Department of Computer Science and Information Systems

A Model for Adaptive Multimodal Mobile Notification

William Brander

Supervisor: Prof. J.L. Wesson

December 2007

Submitted in partial fulfilment of the requirements for the degree of Magister Scientiae in the Faculty of Science at the Nelson Mandela Metropolitan University

Acknowledgements

I would like to thank Prof. Janet Wesson for her invaluable guidance, supervision, patience and advice over the duration of this dissertation. I would especially like to thank her for the long hours she spent reading the many versions of this dissertation and for the useful feedback and criticism that she gave to improve its content and structure.

I would also like to thank all the staff and fellow students in the Department of Computer Science and Information Systems who also provided assistance, as well as Bosasa Operations and On-IT-1 for providing the funding, technology and application domain necessary for this research.

Table of Contents

List of Figures

List of Tables

Summary

Information is useless unless it is used whilst still applicable. Having a system that notifies the user of important messages using the most appropriate medium and device will benefit users that rely on time critical information. There are several existing systems and models for mobile notification as well as for adaptive mobile notification using context awareness. Current models and systems are typically designed for a specific set of mobile devices, modes and services. Communication however, can take place in many different modes, across many different devices and may originate from many different sources.

The aim of this research was to develop a model for adaptive mobile notification using context awareness. An extensive literature study was performed into existing models for adaptive mobile notification systems using context awareness. The literature study identified several potential models but no way to evaluate and compare the models. A set of requirements to evaluate these models was developed and the models were evaluated against these criteria. The model satisfying the most requirements was adapted so as to satisfy the remaining criteria. The proposed model is extensible in terms of the modes, devices and notification sources supported. The proposed model determines the importance of a message, the appropriate device and mode (or modes) of communication based on the user's context, and alerts the user of the message using these modes.

A prototype was developed as a proof-of-concept of the proposed model and evaluated by conducting an extensive field study. The field study highlighted the fact that most users did not choose the most suitable mode for the context during their initial subscription to the service. The field study also showed that more research needs to be done on an appropriate filtering mechanism for notifications. Users found that the notifications became intrusive and less useful the longer they used them.

Keywords: Mobile Notification Systems, Context Awareness, Multimodality, Field Testing, Adaptive Mobile Notification System.

Chapter 1 Introduction

1.1 Background

Computer mediated communication has received a lot of attention over the past few years making it one of the fastest advancing fields in information technology today. One of the areas of computer mediated communication to benefit from the research done in the field is email. Email allows messages to be sent and received almost instantly, regardless of the location or availability of the receiver. Email has been shown to be a rich communication medium, boosting productivity and decision making (Neufeld *et al.* 2001). The convenience and benefits have made people semi-reliant on email, making it one of the most popular means of communication.

The fact that the recipient doesn't have to be physically able to read the message straight away is one of the reasons that the use of email has become so popular, but is also a shortcoming. If the message were to contain time critical information and the recipient were unable to read the message as soon as it arrived, the information sent could potentially become useless. An example is that of the stock market. Prices of stocks fluctuate frequently, and knowing when to buy or sell depends on having up-to-date information about all the factors that could influence the price of stocks. If information is delayed because the stockbroker is out for lunch or for some other reason, large amounts of money could be lost without even knowing about it (Bazinette *et al.* 2001).

A system whereby notification can be sent to a person, regardless of his/her location, once a message of significant importance has been received has many obvious and hidden benefits. The stockbroker leaving for lunch can be notified via his mobile phone (or some other mobile device) and make arrangements for damage control, or in the best-case scenario use the information to make money even if the stockbroker is not physically able to be in his/her office. The fact that the recipient has knowledge of something happening means it can be used to limit losses or make money.

Notification of a received email is just one aspect of email notification that can be useful. The entire message could be sent to the user, depending on the user's devices. Having the message with all of its contents and information will allow for better decisions to be made on the spot.

The usefulness of mobile notification is limited by the capabilities of the mobile device used for that purpose. If a cell-phone is the only device the user is carrying, it is unfeasible to display the entire contents of every message as it is received. Also, if the person is unable to read a textual message for some reason (such as when he/she is driving), he/she should be able to get the information using speech. Thus, for mobile email notification to be of use, multimodality should be considered. It has been shown that multimodal interaction is more effective than single mode interaction, specifically for people in a pedestrian environment (Jöst *et al.* 2005).

1.2 Relevance of Research

Information is useless unless it is used while it is still applicable. Having a system that notifies the user of important emails using the most appropriate medium and device will benefit users that rely on time critical information. There are currently several systems and models for email notification, varying from sending text messages to the users, to sending the header of the email to the user and allowing him/her to respond (Wei *et al.* 2005, Nah *et al.* 2005, Cugola *et al.* 2002, Asaf *et al.* 2000).

Blackberry is one of the technologies currently available for mobile email notification. Blackberry sends the header of the email to a user's Blackberry-enabled cell-phone once it arrives at the server. If the email is important, the user can request the full email and it will then be sent to the user. Since current cell-phone technology already allows for sending emails, the user is then able to respond to the email if required. Blackberry includes several good features that should be included in a mobile email notification system, but it also has shortcomings. The only devices which are supported in any form are Blackberry-enabled mobile devices. No pre-filtering is done on the emails forwarded to the user implying that all emails received,

whether important or not, initiate a notification to the user (Research in Motion, 2005).

Microsoft Office Outlook 2007 includes a mobile notification system, called Outlook Mobile Service (Microsoft, 2007). Outlook Mobile Service supports both text and multimedia messages. When a message is received it is forwarded directly to the recipient's phone. The message is forwarded to the recipient's phone regardless of the context of the recipient.

There are currently several existing models that support some, but not all, of these tasks. Adapting one of these existing models to better suit mobile notification of emails would provide an improved model supporting multiple modes of communication and context awareness.

1.3 Problem Statement

The aim of this research will be to develop a model and a prototype for mobile notification of email using multimodality. Validation of the prototype will also be performed to evaluate the model.

1.4 Aim of Research

The following research objectives were identified:

- An investigation into mobile notification technologies;
- an investigation into displaying email on different mobile devices;
- an investigation into the different types of contexts that will support the selection of the most appropriate mode of notification;
- defining the requirements of mobile notification;
- the design of a model for mobile notification of emails using multimodality;
- the implementation of a prototype to evaluate the proposed model; and
- the evaluation of the prototype and the benefits it provides.

1.5 Scope of research

The scope of the research will be limited to email communication. Emails can contain important information, as well as information with little or no relevance. There are currently only limited models for mobile notification of incoming emails and separate models for filtering of email relevance. The scope of the research is therefore limited to email data and determining its relevance to a user.

1.6 Research Methodology

Effective research needs to have a structure in order to keep it on schedule (Hidding, 1997). Research questions aid research by always keeping a specific goal in mind while performing research. There are various means in which research questions can be answered; these methods include literature studies, arguments, mathematical proofs, prototypes amongst others. Table 1.1 lists the research questions involved in this research, broken down into related groups based on the different chapters of this dissertation.

A literature study will be used to identify existing mobile devices and mobile notification services. The literature study will also facilitate a comparison of existing mobile notification services and context aware mobile notification services. A literature study will also be done in order to research multimodality and its application to mobile devices and notification services. Email notification will also be analysed by means of a literature study.

The proposed model will be designed by taking various features of the models identified by the literature study and combining the positive features of each model in order to address any shortcomings of these models.

A prototype will be implemented in order to demonstrate the effectiveness of the proposed model and to allow a user evaluation to be performed.

A user evaluation will then be performed in order to evaluate the effectiveness of applying multimodality and context awareness to a mobile notification environment.

	Research Question	Research Methods
1	What are mobile notification services? 1.1 What are typical mobile devices? 1.2 What are typical mobile notification services? 1.3 What mobile devices support notification services? 1.4 What existing models are there for mobile notification services? 1.5 What are the limitations of these models?	Literature Study
2	What is multimodality? 2.1 What is multimodality? 2.2 What are multimodal interfaces? 2.3 What are the requirements for a mobile notification service to be multimodal? 2.4 What models are there for mobile notification services using multimodality? 2.5 What is context awareness? 2.6 What models exist for multimodal mobile notification? 2.7 What are the limitations of these models?	Literature Study
3	What are the requirements for email notification? 3.1 How can the relevance of an email be determined? 3.2 How much of the email is relevant to the user?	Literature Study
4	How should the proposed model be designed?	Design
5	How should a prototype be implemented based on the proposed model?	Prototyping
6	What are the results of evaluating the prototype for adaptive mobile notification?	Evaluation
7	What are the results of the research?	Conclusions

Table 1.1: Research questions and methods

1.7 Dissertation Outline

Chapter 1 of this dissertation provides motivation for this research, providing a background to the problem with some idea as to where the research will be heading. The research objectives are stated and a research methodology is given detailing the research questions to be answered.

Chapter 2 consists of an investigation into mobile devices, mobile notification services and existing models. The majority of Chapter 2 is comprised of a literature study, answering questions on mobile technology and support.

Chapter 3 discusses multimodality and context awareness and the benefits of these to mobile notification. An investigation into how context can be incorporated into mobile technologies is also performed. Existing models for multimodality and context awareness are analysed and discussed.

Chapter 4 discusses the design of a model for mobile notification of email using multimodal user interfaces and context awareness. The model is documented and analysed, detailing all the different sections, why they are needed and how they contribute to the system as a whole.

Chapter 5 details the implementation of a prototype based on the proposed model, the problems encountered and any changes that are needed to the model.

Chapter 6 discusses the evaluation of the prototype implemented in Chapter 5 and the results of the evaluation.

Chapter 7 concludes the dissertation by evaluating the achievements and results of the research. This is followed by ideas for future research.

Chapter 2 Mobile notification services

2.1 Introduction

This chapter aims to answer research question 1 in Table 1.1. The first objective of this chapter is to investigate mobile technology, typical mobile devices and mobile notification services. The second objective is to provide a set of criteria to evaluate and compare existing models for mobile notification services. A third objective is to give a brief overview of location awareness. Several existing models are then compared and evaluated based on these criteria and an appropriate model selected to be used for multimodal interfaces and context awareness.

2.2 Mobile technology

Mobile technology consists of devices, infrastructures and protocols that communicate data between systems and individuals in remote locations. This means that any device that can be used anywhere to provide some form of communication, can be classified as a mobile device. Mobile technology is enabling businesses to conduct business more effectively (Nah *et al.* 2005). Mobile technology has impacted on peoples' personal and professional lives and will continue to do so in the future (Ruuska-Kalliokulju *et al.* 2001).

Current trends in mobile technology are non-intrusiveness, multimodality, context awareness and modularity. Four major current research topics in mobile technology are (Ruuska-Kalliokulju *et al.* 2001):

- Context of use;
- personalization;
- applications and services; and
- connectivity of communication appliances.

Mobile technology that takes the context of use into account, can allow the user to personalise the service, provides complete functionality and interacts with other

devices would be ideal. The infancy of mobile technology means that this ideal is still a long way off.

2.3 Mobile devices

The increased interest in mobile computing and applications has led to the rapid advancement of mobile technology. This has created a large pool of mobile devices which can be used. Some of these devices support native collaboration and information exchange between them, whilst others are completely incompatible and need an extra system to allow communication between different devices (Duh *et al.* 2006).

Figure 2.1: Examples of mobile devices

Examples of mobile devices include cell phones, laptops, personal digital assistants (PDAs) and tablet pocket computers (Figure 2.1). Whilst laptops have been in use for a long time, cell phones are still the most popular mobile device in use worldwide with over 1.1 billion cell phone users in 2002 alone (ITU, 2003). This is a direct result of the personal nature of cell phones and the fact that they are relatively inexpensive when compared to other mobile technology. Cell phones are becoming more capable of general purpose activities, making them more like PDAs than before (Ruuska-Kalliokulju *et al.* 2001). The term for this new breed of cell phone is a Smart phone. Smart phones provide tools like calendars, email composers, web browsers and media. Due to the popularity of cell phones, these devices will be used for the notification of email data in our proposed model. This has the benefit of not

requiring users to purchase any additional hardware as most users already have a cell phone.

Communicating with a mobile device depends on the communication methods supported by that device. Cell phones may, for example, receive communication by means of phoning the device, sending a SMS, using GSM or something similar to 3G or UMTS (discussed in the next section). Due to the variations in possible wireless mediums, a modular approach is proposed where a separate module can be used for each wireless medium. This means that as soon as a new medium is developed, it can be included in the model with little change to the underlying model and implementation.

2.4 Wireless mediums

Currently, there are many wireless mediums that can be used for transmitting data to and from mobile devices. These mediums include Ultra Wideband (UWB), Bluetooth, Wireless Fidelity (Wi-Fi), WiMAX, CDMA2000, Wireless Application Protocol (WAP) and General Packet Radio Service (GPRS). In order to determine which medium is best to use for mobile email notifications an evaluation must be performed.

UWB is designed for short-range networks, multiple device connection and high-bandwidth data. UWB works well with other longer range wireless protocols as it can be used as a router for information between local devices when only one of those devices has access to longer range signals.

Bluetooth is a wireless specification for short-range networks and various device connections. Being radio based, it allows a maximum range of 10 metres. Bluetooth allows devices to communicate with each other without the restrictions that other wireless technologies impose (Chozam, 2005). For example, Infrared requires the devices to be ‘looking’ at each other. Depending on the capabilities of the Bluetooth device in question, the user may be able to transfer files, act as a wireless modem, gain local area network (LAN) access or even synchronize data (Bluetooth Special Interest Group, 1999).

Wi-Fi (short for wireless fidelity) is also known as IEEE 802.11b (or 802.11g) which is a standard allowing Wi-Fi enabled devices to interact with other Wi-Fi enabled devices as long as they are both within range of an access point. The Wireless Ethernet Compatibility Alliance (WECA) certify Wi-Fi products that meet their standards, meaning WECA certified products can interact with each other regardless of the manufacturer type.

Worldwide Interoperability for Microwave Access (WiMAX) is also known as IEEE 802.16 specialising in point-to-multipoint broadband wireless access. WiMAX has both longer range and faster speeds than Wi-Fi, but the two are similar in their functionality. Both Wi-FI and WiMAX work without line-of-sight (IEEE, 2005).

General Packet Radio Service (GPRS) is a non-voice service for the sending and receiving of information across a mobile telephone network. GPRS is billed based on the amount of data transferred and not the time taken. Some applications for GPRS are chat, multimedia file transferring, web browsing, document sharing, internet email, remote LAN access and file transfer (GSM World, 2005).

3G is a family of standards using the 1xEV-DO (1x evolution-data only) system, which was designed for burst packet transfer and has a peak data rate of 2.4Mbps with an average data rate of around 600Kbps. Users share the data stream with a time multiplexing of 1.67ms each. 3G is currently being incorporated into most cellular networks worldwide, and because of this, GPRS networks are likely to become obsolete (Yoshimura *et al.* 2004).

GSM (Global System for Mobile communication) is a radio transmission protocol currently used by cell phones to provide data transfer for standard voice phone calls. GSM users can send or receive data at rates of up to 9600bps to a variety of services. GSM is a digital network providing Group 3 facsimile as well as Short Message Services (SMS). GSM is currently being used for voice phone calls on cell phones, but is likely to change with the popularity of Voice Over IP models running on the 3G platform (Scourias, 1997).

Currently, cell phones support the following wireless mediums: GSM, GPRS, 3G and Bluetooth. These mediums will therefore be compared in order to determine the most effective wireless medium for mobile email notification. The following criteria will be used to compare these mediums:

- Data throughput;
- suitability for motion; and
- environment.

	GSM	GPRS	3G	Bluetooth
Standard	GSM	2.5G	3G	802.15.1
Throughput	9600bps	Up to 384Kbps	2.4Mbps	Up to 720Kbps
Suitability	Driving	Driving	Driving	Stationary or slow walking
Environment	Indoors or outdoors	Indoors or outdoors	Indoors or outdoors	Indoors

Table 2.1: Comparison of mobile technologies

Bluetooth is more suitable for a controlled, indoor environment where the device is mostly stationary whilst GSM, 3G and GPRS are suited for applications where the device would be moving around a lot (Table 2.1). Bluetooth has decreased signal strength when the signal has to travel through thick walls with metal shielding, either from cabling or the metal used for structural support. Due to the current popularity of cellular devices, the GSM and GPRS infrastructure is comprehensive in most countries providing a sufficient signal in almost all parts of the world with 3G becoming more and more popular. All of the wireless mediums can be used for mobile email notification, but Bluetooth is limited in the range of use and will therefore not be considered.

2.5 Mobile notification services

Mobile notification can be defined as sending a text or multimedia message to a group of wireless subscribers. Notification is a form of push technology where information is transferred as a result of the occurrence of an event (Cugola *et al.* 2002).

A wireless subscriber is a consumer that subscribes to events in which the user is interested. In other words, the subscriber lets the system know that he/she wants to be informed whenever certain events occur (based on location or some other factors). Once these events occur, the event is published to all consumers who have subscribed to that particular event. An event is published by notifying all subscribed consumers by using any (or the most applicable) device and medium available.

Improvements in the fields of wireless mediums (WiFi, Bluetooth, etc) and mobile devices have given rise to the new paradigm of mobile computing. Cell phones have become the most popular mobile device in use today because of their useful, convenient and personal nature (Keshav, 2005). Since cell phones are primarily communication tools, they provide ideal support for notification across various modes (text, voice or multimedia).

2.5.1 Characteristics

Communication can follow two forms: User-to-User, or User-to-Information (Ghini *et al.* 2000). User-to-User follows closely on collaboration, whilst User-to-Information is more suited to notification (push technology). Due to the nature of mobile devices it is impossible to know the number of devices that will subscribe to events beforehand or to predict whether or not a device will be present when an event occurs or not. To address these issues using a pull-based (the client requests and the server responds) approach requires that a client would have to repeatedly query the server to check for events. This clearly would lead to excessive network traffic and congestion. Also, the nature of mobile devices limits the resources available to them. Pull-based approaches would require extensive searches of the entire network to check for new event sources. On a mobile network (invariably large) this is practically impossible.

A solution to the concerns above can be found using a push-based approach. Push technology consists of a set of consumers (who subscribe to events) and information providers (which listen for the events and then publish them to the subscribed consumers). An event is a message generated by a provider, addressed to the set of consumers with a subscription that matches the event.

2.5.2 Requirements

A number of requirements arise when using push-based mobile applications. Mobile applications are, by their very nature, mobile, and users must be allowed the freedom of moving around without having to worry about whether or not the system caters for mobility. Due to the large number of mobile users in the world, millions of subscriptions must be catered for as well as allowing for many event sources. Each subscriber may also be one or more event publishers. Also, the large number of varied mobile devices implies a need for multiple content formats and mediums (both PDAs and cell phones are mobile devices but do not share a common communication method or media format). The security and privacy of the consumers is important. Allowing users access to other users' information is a matter of privacy invasion, and not all subscribers should be allowed to receive all events that occur (a user should not be able to receive events pertaining to another user without consent).

From these concerns, the following requirements of a push-based system for mobile users can be derived (Table 2.2).

1.	Take the mobility of users into account.
2.	Process multiple content formats.
3.	Support large amounts of subscribers and publishers.
4.	Support security and privacy of information.

Table 2.2: Requirements of a mobile notification service

2.6 Examples of Services

2.6.1 Internet Mail to Short Message Service (SMS)

This example consists of integrating mobile telephony services and networks into a system that alerts users when a new email has arrived at their desktop pc (Figure 2.2). When new mail arrives, the system sends a notification to the user by means of an SMS. Since people sometimes get in excess of over three hundred emails a day (Gopal *et al.* 2001), rules can be created which dictate how the incoming emails are filtered and which emails the user is notified of. The system does not have to send the entire email to the user, and rules can be specified that dictate what part (if any) of the email is sent to the user. The user can also request that the headers of all emails be sent, and then the user can select from a list which of the emails are required. The system can also respond to requests that certain emails get redirected to a fax or, are converted to void and sent with a traditional voice phone call. The other alternative is using the system to compose and send emails. The email message is composed using the cellular telephone and sent back to the system. The system then transcodes the message into the correct format for email and sends it to the intended recipient.

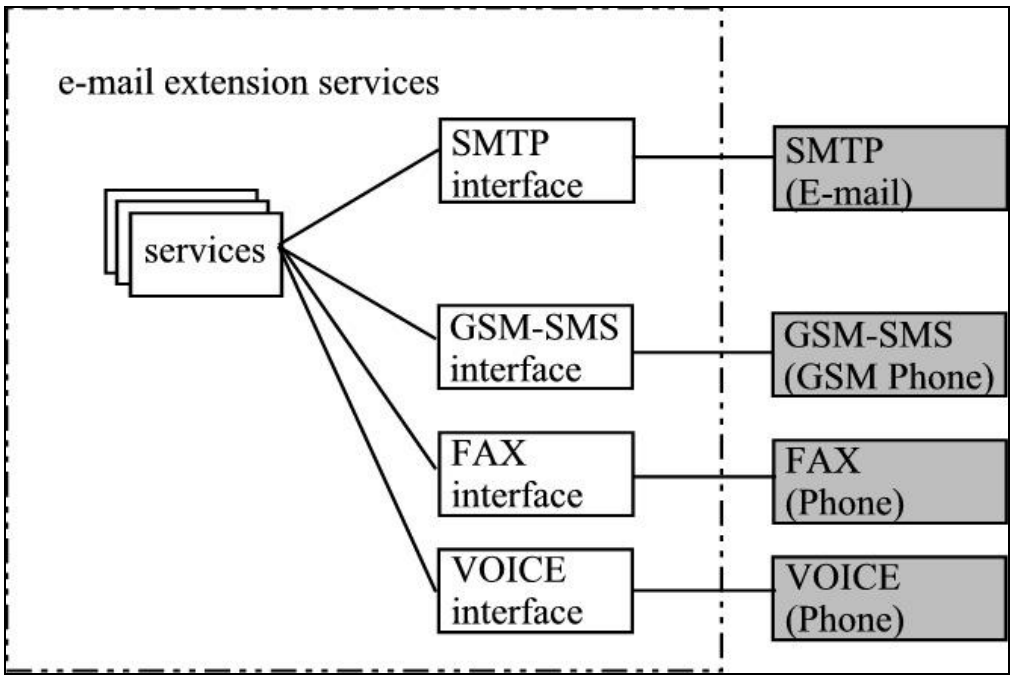

Figure 2.2: Internet Mail to Short Message Service (Ghini *et al.* 2000)

2.6.2 Remote Control and Monitoring of Industrial Processes

In this example, phone features and computer technology are integrated into an industrial environment where processes and security are monitored by a computer system and if the need arises, a system administrator or some other emergency personnel are notified (Figure 2.3). Usually, the security information and the information gathered from the processes being monitored are handled by the computer system, but in extreme cases it may be necessary for a human to intervene in a situation. The administrator can be contacted via different means of notification (mobile or otherwise) including SMS, email, voice phone call or MMS. If the situation requires input from the system administrator, response via SMS or some other sequence of commands from a cell phone could be received by the system. In more difficult situations, such as where a security breach has been detected or a fire outbreak has occurred, emergency services such as the Police or Fire Department may need to be alerted. The system could cater for these situations by means of transcoding an emergency message into voice and sending this using a telephone call.

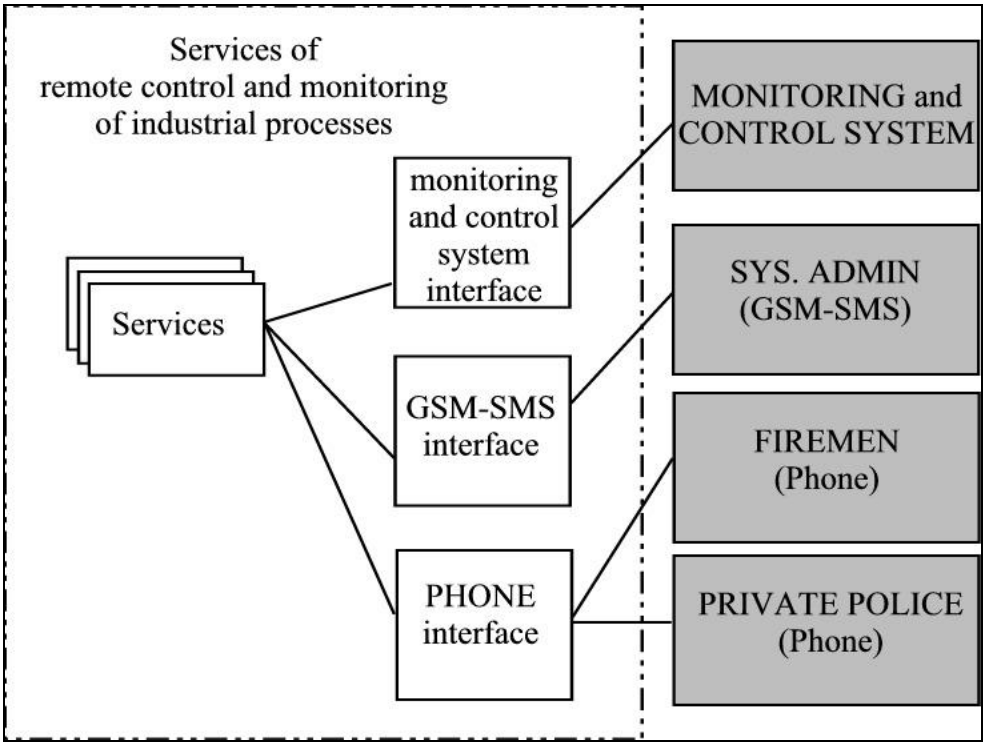

Figure 2.3: Remote Control and Monitoring of Industrial Processes (Ghini *et al.* 2000)

2.6.3 Mobile Notification Services for the Real Estate Industry

Real estate, being mostly location-based, can benefit greatly from the combined fields of mobile notification services and location awareness (Costello, 2001). A system was implemented where estate agents could drive along a route showing houses along the way to prospective buyers. Once the location of the estate agent was determined to be within the range of a particular property, the details concerning the condition, history, price and other facts about that property, were sent to the estate agent. Property details were also sent to the potential buyers for review at a later stage. The estate agent also used the mobile device to request a credit check on the potential buyers whilst showing the property to them at the same time. Any offers placed on the property were sent to the owners as they were made, keeping the sellers informed of what was happening and when (Costello, 2001).

Another very effective application pertains to people who are renting property. At a certain time of the month, if the rent has not been paid, a message can be sent to the tenants reminding them of the payment due. The tenants could also use cell phone banking to pay these bills directly from their mobile devices.

Maps could also be sent to potential buyers if they become lost while trying to find a property. This could also be extended to sending maps and property details to people who have subscribed to a "buyers list" where the subscriber is notified as soon as a new property becomes available.

2.6.4 Comparison of services

The above examples all attempt to provide useful information to users in their current environments. In the first example users are notified as soon as they receive new emails. In the second example users are notified as soon as something requiring human intervention occurs. These examples both notify the users when an external trigger occurs.

The third example is slightly different in that the event triggering the notification is the user. The trigger in this case is the user's location. Location is often used as a means of triggering notifications to users, but is less likely to be used as a means of changing the type of notification being made (Schmidt *et al.* 2006).

2.7 Location awareness

Mobile devices are designed to be used on the move and relieve the restriction of traditional desktop systems of having to be somewhere specific in order to use them. This mobility allows for each mobile device, each mobile service and each mobile application to be used in several different locations. Each of these locations can have their own specific attributes that could potentially be of use to the mobile user or application within that location. The reason location awareness is discussed under mobile notification services and not context awareness, is that location awareness is becoming a standard feature on mobile devices and it is getting difficult to separate the two concepts (Steinfield, 2003).

Many examples can be found in the emerging field of location-based services in mobile commerce (Steinfield, 2003). Location-based advertising is becoming more popular with vendors. Consider that a potential customer is driving (or walking around the inside of a shop, depending on the location techniques used) and the system realises that the customer is close to shop A. Shop A is also having a sale on certain products, some of which match the customer's profile. Upon detecting a match between the customer's location, the customer's profile and the merchandise being advertised, the customer is then sent an advertisement in a particular format suitable for the specific mobile device the customer is using.

Other examples of location-based services could be for parents to have knowledge of where their children are at all times. Many parents spend time worrying about the safety of their children. Equipping their children with location-aware mobile devices enables the parents to know exactly where their children are at all times.

There are three broad categories of location awareness (Steinfield, 2003), namely:

- Either the network or some global system calculates the device's location (Method 1);
- the device itself is aware of its location (Method 2); or
- a hybrid approach between the two is used (Method 3).

There are, of course, many other approaches to determining the location of a mobile device, but a comprehensive study is beyond the scope of this research. As such, only a brief overview of the three methods will be given.

In the case of the network calculating each device's location (Method 1), the location can be found either by using the cell of origin where the location of the nearest signal tower is given as the device's location. Alternatively, calculating the Time of Arrival triangulates the location of the subscriber using timing signals between at least three signal sources (Djuknic *et al.* 2003). The Federal Communications Commission (FCC) has issued a mandate called the Enhanced 911 (E911) mandate stating that all wireless operators have to be able to provide the location of a device automatically to public safety answering points such as the police or other emergency rescue services (FCC, 2003).

In order for a device to know its own location (Method 2), it has to have some means of comparing signal strengths between different sources or some other form of distance measurement. One of the most well known of these methods is GPS, which uses triangulation of four satellites out of a possible 24 (Djuknic *et al.* 2003). Another possible means for a mobile device to know its location is to use the timing signals sent from multiple signal sources and to triangulate its position from that data.

Hybrid approaches (Method 3) involve multiple location detection techniques in order to best determine the device's location and improve accuracy (Nord *et al.* 2002). Assisted GPS (A-GPS) embeds GPS receivers in a cellular network (Wright *et al.* 2001). This helps reduce the calculation burden and delay when requesting position. A summary of the strengths and weaknesses of each approach is given in Table 2.2.

Type	Method	Strengths	Weaknesses
GPS	1	Highly accurate. User privacy.	Loss of accuracy in "urban canyons" and indoors Delay when requesting position Large size
Observed Time Difference	1	Low cost Low complexity	Network needs to invest Modification required to handset
Cell ID	2	Low cost No modification necessary	Low accuracy in rural areas Loss of privacy
Time of Arrival	2	High accuracy Can determine velocity as well No modification necessary	Loss of privacy New equipment needed by network providers
Assisted GPS	3	Less delay than GPS Lower cost	New handset required
Multiple location sources	3	Increased accuracy Greater flexibility More robust	Loss of privacy High cost

Table 2.3: Comparison of location detection services

The scope of this research project is limited to cell phones. Since not all phones support GPS, determining a user's location using GPS may not be the most suitable method. The same applies to using Assisted GPS and the Observed Time Difference methods. The most suitable method would be to use either the Cell ID or the Time of Arrival for determining the user's location. This method does not require additional handsets and is a relatively low cost solution.

2.8 Existing models

There are many models that already provide mobile notification services suitable for email notification, but not all of them satisfy the requirements outlined in Table 2.2. Some of these models include location awareness, but not all of them. The two concepts can exist independently of each other. A discussion of some of the existing models is given in this section and a short comparison between them is used to highlight the similarities between these models.

2.8.1 A General-Purpose Location-Based Notification Service

The General-Purpose Location-Based Notification Service (Oh *et al.* 2003) model was designed to create a location-based notification service which could be easily applied to a number of situations without modification. A number of factors were considered when creating the model, such as flexibility, robustness and the speed or responsiveness of the model. It was argued that current architectures cannot handle the load of all the positioning requests and a new approach was needed. With this in mind, the authors created a model that supported the following features (Oh *et al.* 2003):

1. It can locate subscribers according to the FCC E911 mandate, and can also be extended to use automobile navigation systems for tracking;
2. it allows for precisely defined location notification areas so that any area of interest can be added;
3. the system is aware of a subscriber entering a notification area within one minute of their entrance; and
4. it has the ability to track subscribers such that condition 3 is satisfied.

This model (Figure 2.5) consists of defining the notification areas, detecting subscribers within a notification area and composing and delivering the notifications to the subscribers. Each of these components is discussed in more detail, explaining the principles behind each of them.

Defining a notification area is accomplished by the publisher (or person defining the notification area) creating a Notification Campaign Specification. This Notification

Campaign Specification contains the area of interest, what type of subscriber should receive notification in the area as well as the type of notification that should be sent (SMS or WAP push, or other). The notification service sends the Campaign ID as well as the area constraints to all signal points within the specified area. Each signal point stores the Campaign ID as well as the area information. If a signal point is not within the specified area, then there is no need for it to have knowledge of the notification area.

Detecting when a subscriber enters a notification area is based on the assumption that more people will be eligible to receive the notification than those that are not (Figure 2.4). The location request is initiated on the signal point level, rather than at the network level. This works well with the fact that each signal point knows what areas and type of subscribers it must detect. The proximity detection unit keeps track of the areas of interest and is included as an extra to a normal signal broadcaster. This removes the need for network involvement or for some other server application to monitor everything.

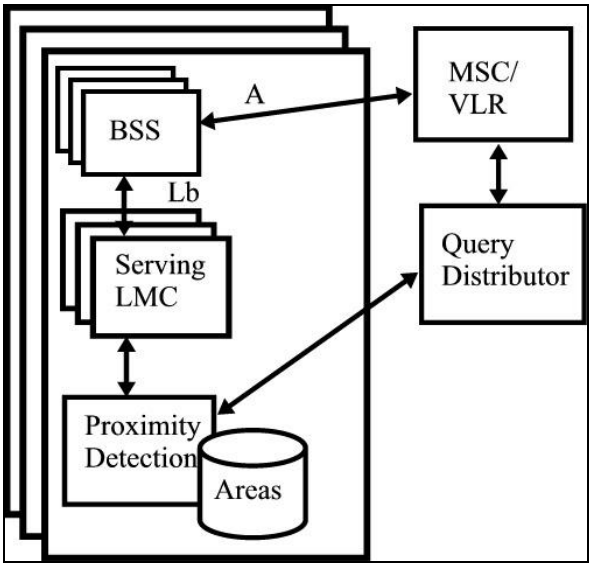

Figure 2.4: Model for High-Rate Proximity Detection (Munson *et al.* 2002)

Composing and delivering the message is the responsibility of the Notification Provider (Figure 2.5). Once a subscriber has entered a notification area, the Notification Provider checks to see whether or not that subscriber's profile matches the profile definition of the campaign. If the subscriber is eligible to receive a

notification, the Notification Provider checks which delivery method to use (WAP, SMS or other). The campaign specification also includes the content, which may change for each message format, of the notification message.

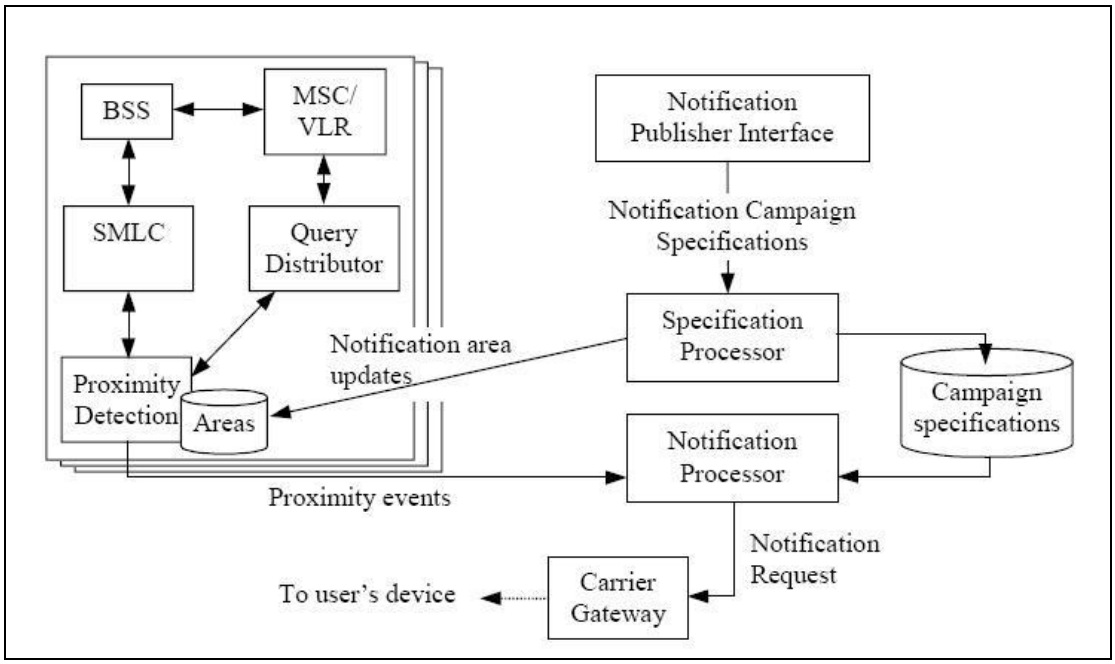

Figure 2.5: A general-purpose location-based notification service (Munson *et al.* 2002)

Evaluating this model against the requirements outlined for mobile notification services (Table 2.2), we see that the model:

1. Supports the mobility of its users;
2. allows for multiple (SMS, WAP push, or other) content formats; and
3. supports large amounts of subscribers.

We can also see that the model provides no additional security or privacy functions other than the security in place with the operators' network. This may be sufficient in some cases, but not all applications of the model will run across secure networks.

Since this model does not support any non-standard security or privacy features, it cannot be easily extended to multiple wireless mediums as some mediums provide no built in security features (e.g. Bluetooth and Wi-Fi). The model also makes an assumption that there will be more subscribers eligible to receive notification than those that are ineligible. Whilst this may be a valid assumption in certain instances,

this is not always the case. As an example, more people will subscribe to traffic and weather notifications as compared to those that will subscribe to commercial notifications (Oh *et al.* 2003).

2.8.2 Dynamic channel model

The Dynamic Channel Model (Afonso *et al.* 2004) is an information push model for notification to mobile users. The model implements the publisher/subscriber model where a publisher is a user sending information and a subscriber is a user receiving information. The model was inspired by the idea of information channels (Cheriton, 1992) where an information channel (or simply a channel) is defined as an abstraction of communication and system resources necessary to distribute information to a set of users. Publishers can use channels to send information (notifications) to a potentially large set of subscribers. A channel can also have its subscribers changing constantly.

All notifications sent across a channel are organised into groups. When subscribers add a particular channel to their reception list, they also specify which groups of information they would like to receive. In this way, each channel can also be split into multiple sub-channels.

Figure 2.6 shows the architecture of the Dynamic Channel Model. The architecture is logically split into a subscriber and a publisher side. On the subscriber side, notification is initiated by a push message to a cell phone. Once the message is received and read, an internet connection is established on the laptop using the cell phone as a GPRS modem. The software on the laptop creates a connection to the web site where the information is hosted and downloads the appropriate information displaying it as a notification for the user. Since the information is stored as a web page, it can be displayed in a variety of formats (including audio or video). The laptop also contains a GPS tracking device, and the device's position is passed through to the location proxy (see Figure 2.7).

The implementation of the model could include security features such as a simple login to more advanced features requiring more security. The location information is

only sent to the server once a connection has been established (after login) and thus the privacy of the user is maintained.

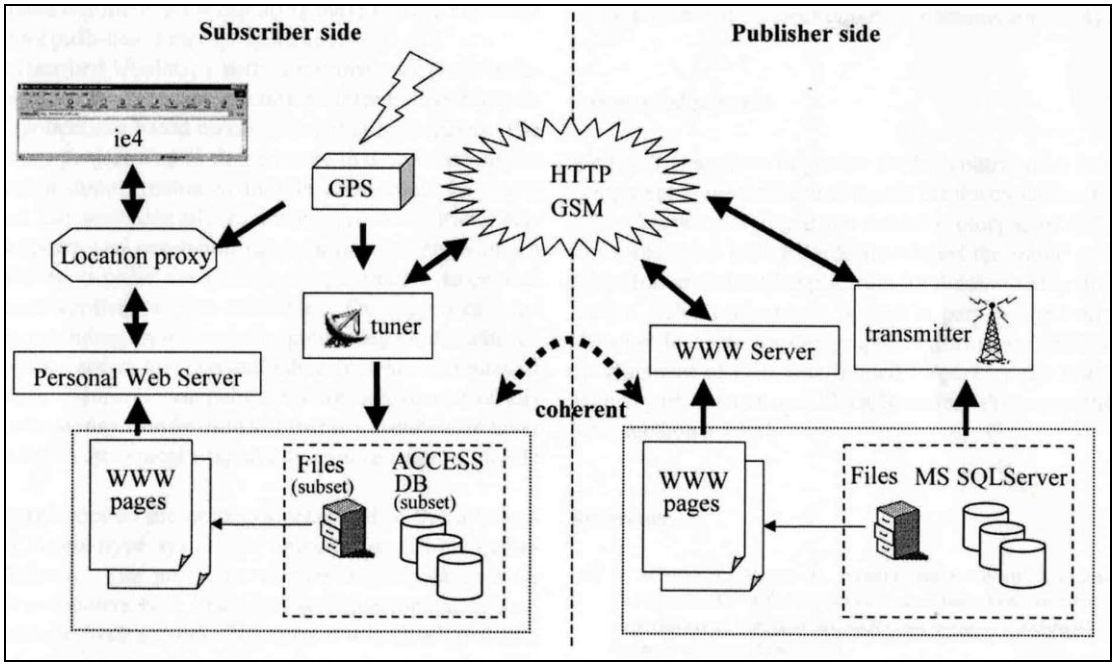

Figure 2.6: Dynamic Channel Model (Afonso *et al.* 2004)

An important advantage of this model is that once the information has been viewed, it is on the mobile device and can be viewed regardless of the network connection status.

When a publisher needs to send information to a set of subscribers, the cell phone transmitter is alerted that it needs to send a GSM message to the set of cell phone numbers. At the same time, the WWW Server hosts the information to be distributed at the Uniform Resource Locator (URL) specified in the GSM message (Figure 2.7). All information sent is stored in a database to allow for upgrades and consistency of information if switching or using multiple providers.

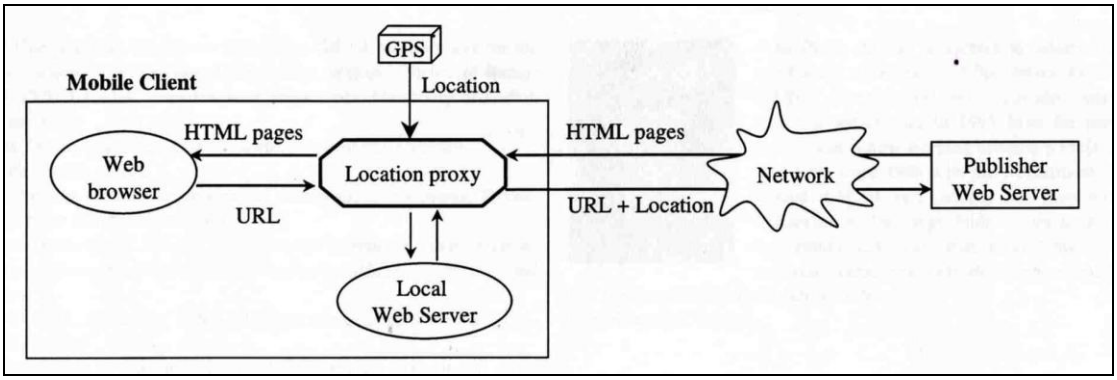

Figure 2.7: Mobile client proxy (Afonso *et al.* 2004)

The model requires that subscribers carry a lot of extra technology with them, namely:

- A cell phone;
- a laptop; and
- a GPS tracker for the laptop.

Other difficulties experienced were the connection speed of using cell phones as modems. With the advent of 3G and GPRS technology, however, this may not be a problem anymore.

Evaluating this model against our requirements (Table 2.2) shows that the model supports:

- Mobility of subscribers;
- multiple content format;
- large amounts of publishers and subscribers; and
- security and privacy.

This model meets all of the requirements for a mobile notification system, but has excessive hardware requirements that not all users will have. The model also uses two separate wireless mediums (GPRS for internet and GSM for initial notification) which increase the areas where problems could arise. Overall, even though the model satisfies all the requirements for a mobile notification service, it is not practical for a general notification service as not all users will have both a laptop and GPRS-enabled cell phone with them at all times.

2.8.3 Multimedia Alerting and Notification Service for Mobile Users

The Multimedia Alerting and Notification Service for Mobile Users model (Wei *et al.* 2004) provides a platform for sending multimedia messages to mobile clients. The scope of the research is limited to email notifications but a multimedia notification system is still applicable as email may be one of the multimedia notification types made. Many new mobile devices support streaming of audio and video and this model takes advantage of these. The model is comprised of three components: the Enterprise Messaging Network (EMN), the Alerting Platform (AP) and the Multimedia Platform. These three components each perform a different task and when combined, integrate to form the complete model.

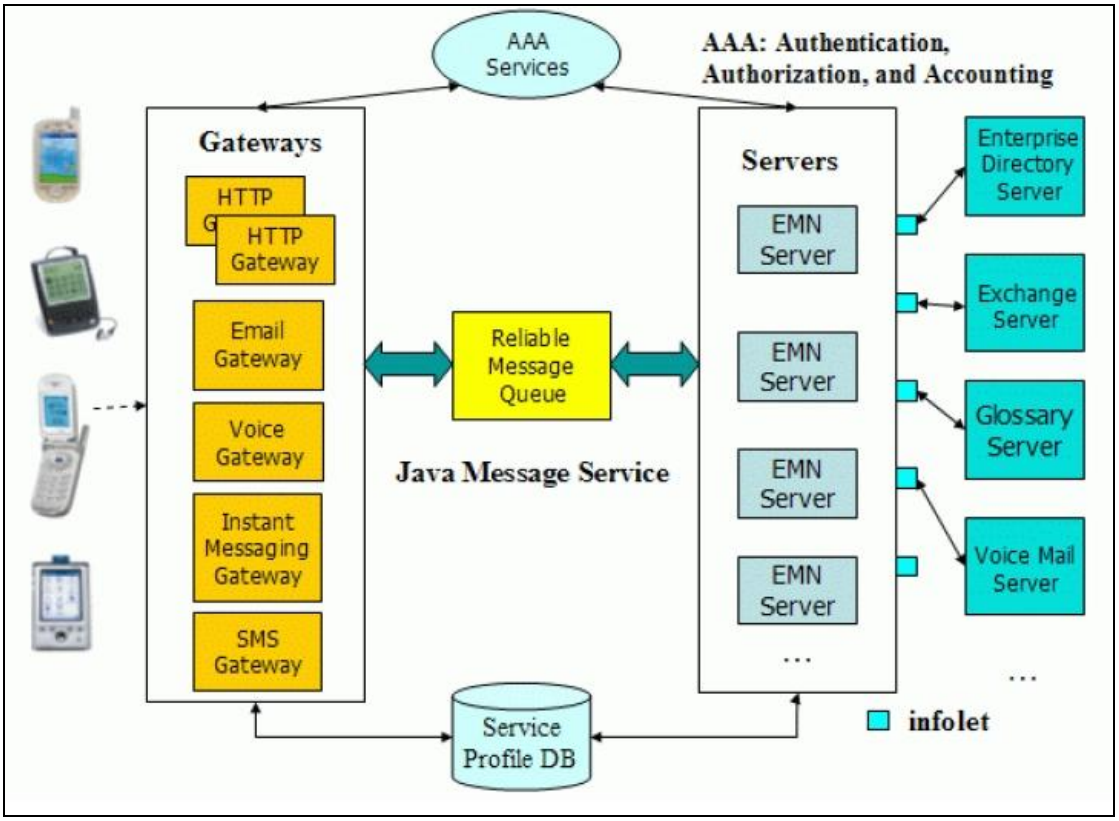

Figure 2.8: Enterprise Messaging Network Service Platform (Wei *et al.* 2004)

The EMN consists of servers, gateways, message queues and a database to store application-related data (Figure 2.8). Each gateway (called "devlets") corresponds to a set of mobile devices, these gateways are the interaction between the model and the mobile devices. The gateways also provide user authentication verifying that the information is going to the right person. The publishers interact with the model via EMN Servers (called "servlets") which in turn interact with the messaging queue in

order to send the notification. If there is any application specific data to be sent, the servlets store it in the database and the devlets read it again once it is their turn to transmit.

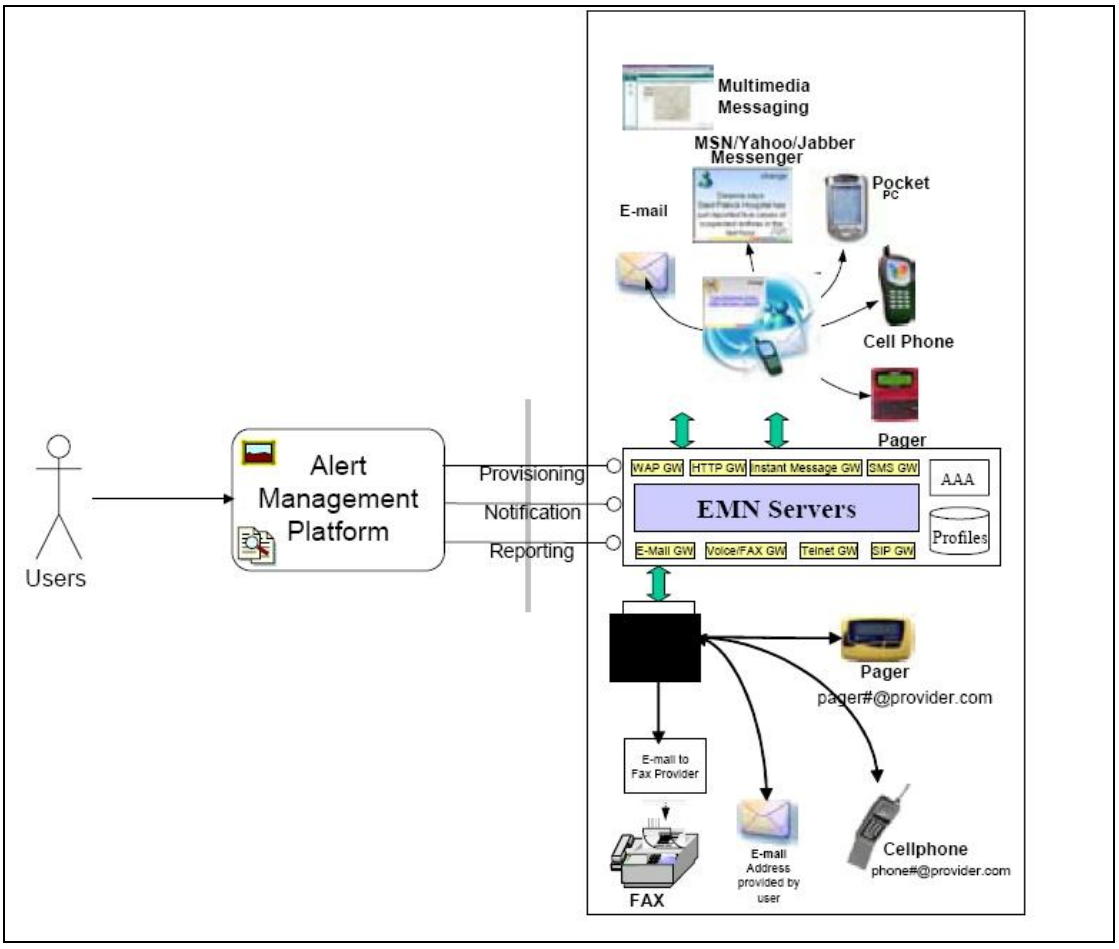

Figure 2.9: Alert Management Platform and Alert Dissemination Engine (Wei *et al.* 2004)

The AP consists of an Alert Management Platform (AMP) and an Alert Dissemination Engine (ADE). Figure 2.9 shows an instance of an alerting platform with a ADE built using an instance of a customized EMN. If a user wants to send a new notification, he/she interacts directly with the AMP, which interacts with the ADE to disseminate the message to all subscribers. The ADE then uses an instance of the EMN to notify all subscribers of the message.

The multimedia platform is responsible for hosting, storing, collecting and transcoding of all multimedia to be sent. The media manager consists of a content manager and a delivery manager. The content manager is responsible for video

storage, acquisition, transcoding and selection. Another of the content managers responsibilities is content addressing, where it must generate the access information for the mobile users to let them know from where to stream the video. The actual streaming is performed by the delivery manager. The content manager interacts with the mobile devices through the EMN, whilst the delivery manager interacts directly with the mobile users. This separation of content from delivery helps address scalability, as there can be many instances of delivery managers and only one content manager, helping to decrease network delay for streaming particularly large video files.

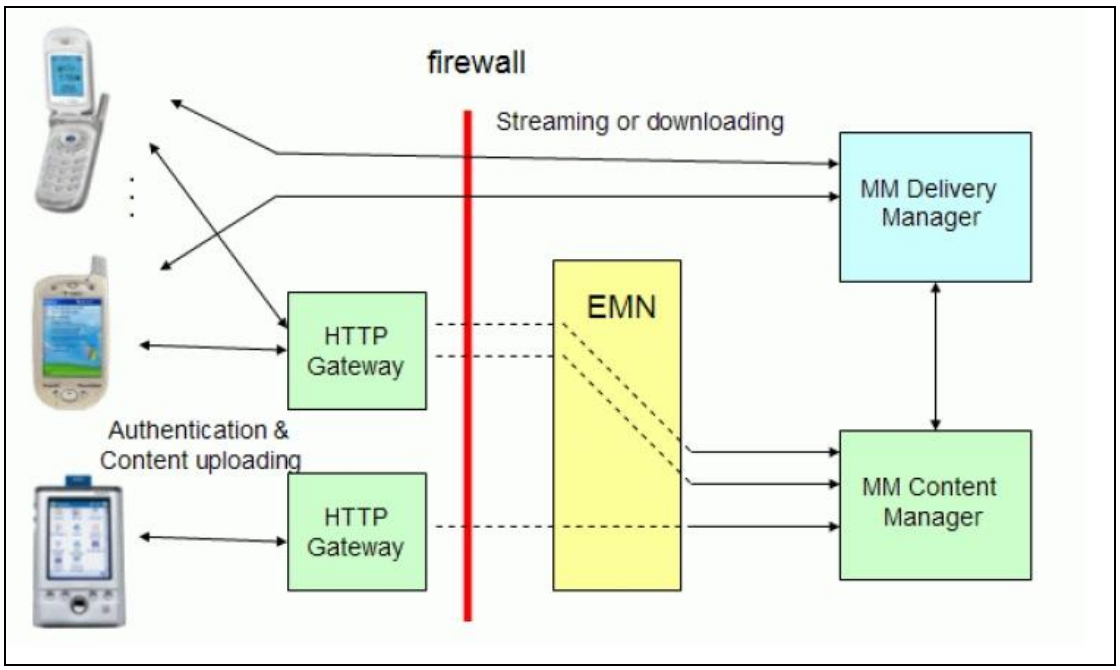

Figure 2.10: System architecture for multimedia services (Wei *et al.* 2004)

The EMN meets the following requirements for a mobile notification service (Table 2.2):

- It takes the mobility of its users into account;
- it allows for multiple content formats;
- it allows for large amounts of subscribers; and
- it supports the privacy and security of its users.

As seen above, the model satisfies all the requirements for a mobile notification system and does not require any additional hardware for the user to carry around other than a smart phone. The model uses two communication channels; one for the initial notification, and another content server for the video or other multimedia. This increases the chances of something going wrong; either of the streams could fail and the users could still assume that the service was online because one of the streams is working, unaware of the fact that there are multiple streams.

2.9 Comparison of existing models

Table 2.4 provides a comparison of the three models discussed above. The models are compared in terms of the requirements they satisfy (mobility, scalability, flexibility and privacy) as well as the convenience of the model (e.g. how much additional hardware is needed).

	Factor	Oh *et al.*	Afonso *et al.*	Wei *et al.*
1	**Mobility**	✓	✓	✓
2	**Scalability**	✓	✓	✓
3	**Flexibility**	✓	✓	✓
4	**Privacy**	✗	✓	✓
5	**Convenience**	• Requires additional hardware for the signal points • Additional privacy scripts needed	• The subscriber needs to carry extra hardware • Time delay when requesting position (due to GPS) • Two separate data streams • Need a web server and a mobile server	• Two separate data streams • Need a web server and a mobile server

Table 2.4: Comparison of existing models

Based on this comparison, the first model does not satisfy the requirements for a mobile notification service and should be rejected. The other two models satisfy all the requirements, but the second model has additional hardware requirements for the user before it can perform effective notification. The second and third models both use multiple streams for sending information. The reason for this is that the information being streamed is not restricted to text only and they were designed prior to (or without incorporating) multimedia messaging technology (MMS's). With these new technologies, the stream requirements could be eliminated from both models. The second model's dependency on extra hardware, leads to the third model being the better model for users. The third model is therefore the model that will be extended to include multimodality and context awareness (Chapter 3).

2.10 Conclusions

This chapter has answered research question 1 in Table 1.1. The first objective of this chapter was to investigate mobile technology, typical mobile devices and mobile notification services. An introduction to mobile technology was given, covering wireless mediums, mobile devices, location awareness and models for mobile

notification. Cell phones were identified as the most suitable devices for mobile notification services due to their pervasiveness and social acceptance (Section 2.2 – 2.4). The nature of cell phones limits the possible notification modes to text, voice or multimedia (Section 2.5).

Location awareness was identified as an important component of mobile notification services. Notifying users of content applicable to their current location was shown to be more effective than notifying users of content which may not be applicable in their current surroundings (Section 2.6).

The second objective was to provide a set of criteria for the evaluation and comparison of existing models for mobile notification services. These requirements were listed and several models were evaluated against these requirements. The Multimedia Altering and Notification Service for Mobile Users model by Wei *et al.* satisfies most of the requirements and was identified as the most suitable model to extend and include multimodal notification (Section 2.9).

In the next chapter we will introduce the concepts of context awareness and multimodal interfaces. We will also look at a few existing mobile notification models that incorporate multimodal interfaces and context awareness.

Chapter 3 Multimodality and Context Awareness

3.1 Introduction

This chapter aims to answer research question 2 in Table 1.1. An explanation of multimodality will be given, followed by a discussion of the relevance of multimodality for mobile notification services. Multimodal interfaces for small screen devices will be discussed and tools and techniques for summarising data will be presented. Several existing models for mobile notification services that incorporate multimodal interfaces will be discussed and compared.

3.2 Multimodality

People communicate using various modes of communication (Brewster, 2002). For example, when you are reading a book, the book is communicating with you using a visual mode of communication. The five senses contribute greatly to which mode of communication is being used. A deaf person will find it difficult to communicate using an aural mode of communication only. Whilst the deaf person may be able to communicate with a hearing person, the hearing person will not be able to communicate back to the deaf person without resorting to some other mode of communication.

Multimodal interaction can be defined by breaking multimodality into its separate components (Nigay and Coutaz, 1993): Multi, meaning many; and modal, containing both mode and modality. The mode refers to how the information is interpreted and modality refers to how the information is presented.

According to our general definition, two people that are talking use voice as the modality of communication and audio as the mode of communication. This is not a complete representation of their interaction, however, as people use other means to communicate, such as visual clues as to their emotional state and aural clues as to their intentions (Marti, 2002). As an example, nervousness can be seen by people

fidgeting with a pen or some other object whilst speaking to a superior; also, people that lie tend to have certain visual and aural clues as to the truth of their words.

This shows that communication is rarely based on single modes or modalities of communication only. Multimodal systems have two aspects that need to be kept in mind during the design process (Nigay and Coutaz, 1993):

1. The combinations of different types of data between different devices; and
2. the temporal constraints imposed on information processing between different devices.

A specific data type can be related to a specific modality. Temporal use is broken into parallel use and sequential use, whilst the various combinations of types of data can be either combined or independent. Parallel use implies that various modalities can be used simultaneously for input and output. Sequential use is using various modalities to achieve a set goal, but the modalities are used one after another. Modalities can also be combined or be independent of one another to achieve the same effect. This is known as fusion. These results can be summarised in Figure 3.1

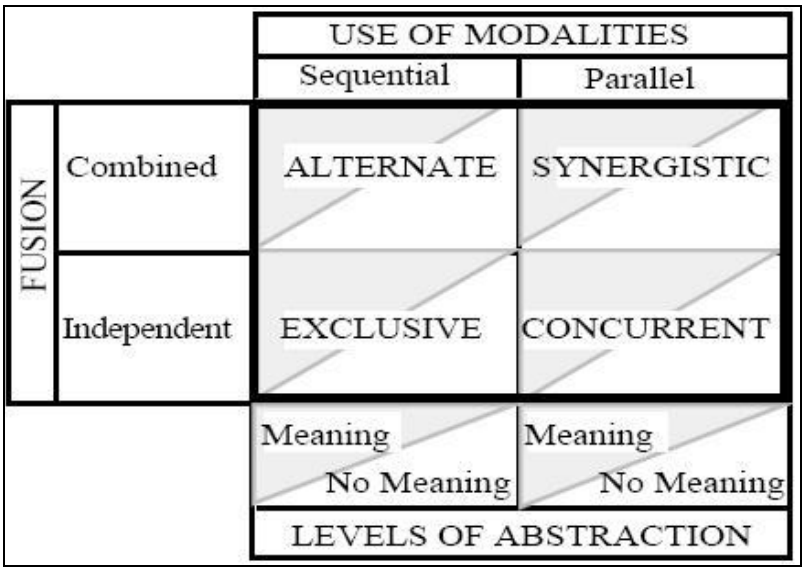

Figure 3.1: Temporal and fusion use of modalities (Nigay and Coutaz, 1993)

Computers are currently able to take advantage of various modalities for communication. Programs designed to aid the visually impaired turn text on the screen (a visual modality of communication) into voice and read the contents (an

audio mode of communication) to the user. The information presented, has thus been converted into multiple modes of communication, hence the term: Multimodality.

Multimodality does not just refer to information presentation to the user. User input can also contain multiple modes of communication (be multimodal). Speech recognition has been used in conjunction with a mouse to achieve a multimodal user interface. Multimodal interfaces are designed to help make user input easier and, in some cases, make interaction possible where interaction would normally be impossible (bright lighting makes it difficult to read a cell phone display screen).

Within the scope of this research, the definition of multimodality is far less general and more specific to the needs of mobile email notification. A multimodal mobile email notification system can be defined as a mobile email notification system that uses multiple sequential modalities to notify the user of any received emails.

3.2.1 Relevance

There are several factors driving the need for multimodal mobile services. Road safety is a major concern since the advent of mobile phones. Driving and using a cell phone without a hands-free kit is illegal in many countries (Cellular-news, 2005), and while there are no laws regarding the reading of a message whilst driving, it is difficult to focus on the road and the cell phone at the same time. Multimodality can provide a means for users on the road to read messages without using any visual senses that will detract from the drivers' attention to the road.

Safety is not the only driving factor for the use of multimodality in mobile environments. The number of mobile professionals in the United States alone (users who spend more than 20% of their time on the road) is expected to be 27.7 million by 2008 (DataMonitor, 2006). Keeping up to date with information is essential for their survival and since they spend so much time on the road, they are the ideal candidates for multimodal mobile services (Colby, 2002).

The driving factors are not purely market based. Multimodal interfaces allow for a flexible choice of input options (Oviatt, 2002). This allows users the choice of what

mode is better suited for what data type. This is good from a usability viewpoint as each user has different opinions when it comes to how user interfaces should respond to user input. The increasing complexity of systems also means that unimodal interfaces are struggling to cope with the myriad of input options available at any one point of system operation (Oviatt, 2002). Multiple modes will allow all users to interact effectively with ease.

Multimodal systems also address various accessibility issues for impaired users (Marti, 2002). Having a visual or aural disability can be overcome with ease by switching to a different modality. A blind person with impaired motor skills can use speech and audio for input and output with a system, while a deaf person can use the visual and tactile modes for using the same system.

Mobile environments can also benefit from multimodality in that multimodal interfaces provide the necessary adaptability to operate effectively in the changing conditions in which a mobile device may be used (Oviatt, 2002). Switching to an audio mode of communication when lighting does not allow for visual methods can aid the usability of the system and allow for use in more locations than the unimodal version of the same system would allow. Movement has also been shown to produce negative results for the effectiveness of small screen displays. Fast movement makes it difficult to operate a mobile device using a visual mode of communication (Brewster, 2002). Using a different mode of communication for this situation has been shown to overcome the problem with using mobile devices on the move.

Another drive for multimodality in a mobile environment is the ease of use it can bring to mobile applications. Mobile applications are not yet mass adopted simply because of the ease of use factor (Brewster, 2005). Multimodality has the potential to enhance the usability of mobile applications by changing the way users interact with mobile devices. Interaction with mobile devices is traditionally based on text-based menus where the user selects an option from a list. This has been shown to be an inadequate interaction paradigm for mobile devices (Brewster, 2005).

User acceptance of multimodal mobile services has been shown to be favourable (Jost *et al.* 2005) with users performing set tasks with greater speed and satisfaction. The

robustness of multimodal mobile interfaces has also been shown to be greater than that of unimodal interfaces (Oviatt, 2002). All these studies show that multimodal interfaces are better suited to a mobile environment than traditional interaction paradigms.

Multimodal systems have also been shown to provide higher levels of user satisfaction (Oviatt, 2000). Users have shown a strong pull towards using multimodal interfaces over their unimodal counterparts. In some studies, it has been shown that simply having to option of switching modalities was enough to provide greater satisfaction even if the user never switched modes (Brewster, 2002). Just having the option was enough to encourage use. User efficiency has also been shown to increase when using a multimodal interface (Oviatt, 2000). Increases in speed of up to 400% have been shown for certain application domains, with the worst case scenario being no increase in efficiency but no decrease in efficiency either (Cohen *et al.* 2000).

Multimodal interfaces also support mutual disambiguation. Mutual disambiguation uses multiple modes of input to validate the accuracy of each recognised input. This provides greater accuracy, especially with regard to speech interfaces (Oviatt, 2000). An example would be if the user says the plural of a word, but the system recognises only the singular. The system could then realise that the plural of the word was meant by the fact that multiple parameters were provided as a list in a second mode.

3.2.2 Multimodal interfaces

Multimodal systems have many options available to them with regards to user input and output. Any combination of modes can be used to achieve a goal as long as the technology is there to support it. Some of the input and output modes currently available are discussed and explained here.

3.2.3 Input

User input is a difficult task in multimodal interfaces. Input from all supported modes at any point in the program must be catered for, and as a result, the design phase of multimodal systems is complex (Oviatt, 2002).

Input in a multimodal system can either be active or passive. Active inputs are those that the user actively selects whilst passive inputs are those that the system collects without the users' knowledge. For example, if the user is using a mouse to interact with the system; this is an active mode of input but at the same time the system could be collecting temperature information which represents a passive input source.

Using information collected by passive sources to alter the meaning of the data is also known as *context awareness*. A context aware application takes the users' environment into account and uses that information to alter the way the data is presented, or even alter the presented data. A good example in a mobile environment would be in a situation with very bright lighting. Bright lighting makes it difficult to read the display of the mobile device, so the information could be presented using sound instead of a visual display.

Location awareness was discussed in Section 2.6. Location is a form of passive input that can be used in context awareness; taking information from an input source without the users' direct involvement and applying it to find out what data is relevant to the user. Location awareness is also an example of context awareness, using only the users' location to bring meaning to the data. Context awareness is much broader than this and may take into account other factors, such as the temperature of the environment, the surrounding noise levels or light levels. While it is relatively easy to design a device to measure all these environmental effects, it is, by comparison, more difficult to derive meaning from these measurements (Koleva *et al.* 2001). More on context awareness is presented in Section 3.3.

Traditional input options are also a part of multimodal systems. They are simply another means of getting information from the user into the system. These options could be a keyboard, a mouse, touchpad or stylus. All of these input devices provide active modes of input. While these are common and users are, for the most part, comfortable with them, they do not provide the required flexibility that most user interfaces require (Jöst *et al.* 2005).

Speech-based input is one of the most common forms of input when multimodal interfaces are discussed. Speech-based input has great potential, but is currently prone to recognition errors, especially when the user has an accent. User testing shows very positive results for speech input (Brewster, 2005). Speech is an example of an active mode of input.

Other modes of input are less obvious than speech, but possibly more effective. Gestures are a good means for input as they can be used without visual attention (Brewster, 2005). Gestures can be performed by various parts of the body, allowing for people with disabilities. Fingers, arms, legs, and even heads can be used for gestured interaction. Using the head for gestured input has been shown to be very effective, especially when combined with a visual mode for output (Oviatt and Cohen, 2000). Gestured input can be either active or passive. If the user responds to the system by purposefully shaking a tactile sensor, that is an active response. If there is a sensor on the user's chair that lets the system know when the user is at his/her workstation; the input is passive.

Devices equipped with accelerometers (detects the movement and orientation of the device it is attached to) can provide a rich means of both active and passive input. Accelerometers have been used to detect muscle tremors associated with the squeezing of an object for input (Brewster, 2005). Having a mouse equipped with an accelerometer allows an extra input source other than the usual movement and clicks on a mouse. The orientation of the mouse can be used, so rotating the mouse will produce a different input from simply moving and clicking with the mouse. Squeezing the mouse can also produce an input source, as this can be detected by the muscle tremors associated with squeezing.

3.2.4 Output

Many existing systems use multiple modes for output. These systems are usually termed "multimedia applications." These usually combine audio and video to output the required information. There are several output methods that are not reliant on audio or video (Oviatt and Cohen, 2000) and these are covered later in this section.

Audio is a very effective mode of communication (Brewster, 2002). There are many reasons for using sound in human-computer interaction (HCI):

- Hearing does not depend on vision. This means that the user can focus his/her eyes on something else while still interacting with a system.
- The amount of information needed on the screen can be reduced by representing some of the information with sound.
- The load on the user's visual system is reduced. Along with less information, there is less for the user to process with his/her visual sense. The amount of information shown on a user's visual displays can lead to the user missing important information.
- The auditory sense is currently underused in almost all aspects of HCI (Bly, 1982). The auditory sense is capable of hearing and comprehending complex sounds, which can each be given a meaning.
- Sound is attention grabbing (Bly, 1982). It is easier to avoid seeing something than it is to avoid hearing something.
- Sound can make computers more usable for visually disabled users by providing a mode of interaction independent of their disability.

From the above, it may appear that sound is a perfect medium for interaction, but it is not without its own flaws, including:

- Sound has a very low resolution. With vision, it is possible to distinguish between colours, size, shape and multiple other features of an object. With sound, only the volume and frequency can be changed (Buxton *et al.* 1991).
- Changing the frequency of a note may change its perceived volume (Brewster, 2002), causing difficulty when using multiple changes to sound to reflect changes in the interface.
- Hearing speech is slower than reading text. It is also difficult to review or preview speech (Brewster, 2002).
- Many people find sound in user interfaces annoying and would resort to turning it off unless they absolutely had to use it (Brewster, 2002).

Speech is an example of using sound for output. The system "speaks" the output to the user; who can then simply listen to it and react accordingly. There has been much progress made in synthetic voice systems and they can even accommodate for noisy surroundings in ways other than simply increasing the volume (Brewster, 2002).

There are also many ways in which sound can be used without speech for output. Non-speech audio is underused in user interfaces and is perhaps a better solution than speech for audio output (Brewster, 2002). The sounds that can be used include music, sound effects and sounds from our everyday environment. With non-speech sounds, the sounds can be shorter than speech, but the user would have to learn the meaning of the sounds (unlike speech, where the meaning is contained in the sound). Non-speech sounds are good at giving status information, showing trends and representing simple hierarchical structures (Brewster, 2002).

3.2.5 Multimodal mobile interfaces

Multimodal interfaces can consist of multimodal input, multimodal output or both. The same applies to mobile environments. Multimodal mobile services must provide a means for user input or output, using more than one mode. Due to the nature of cell phones, two modes of input and output are readily available, namely audio, and video.

The design and purpose of cell phones allows for voice communication (the original purpose of cell phones) and text or image (and more recently, video) communication (via the small screen display) without any modifications or additional hardware.

Multimodal mobile interfaces help address many issues that unimodal mobile interfaces have. Robustness is a particularly important factor in user interface design. If the interface is not robust, how can it still be usable? Multimodality helps achieve a greater level of robustness, particularly in a mobile environment, that would otherwise be difficult to achieve (Oviatt, 2002). It is possible to use a visual mode to display a list of choices and an audio mode to read which item is selected. These two modes can be used to complement each other by providing not only a means for the user to read the list, but also to listen to the list in case he/she is unable to divert his/her eyes from the current task. If the audio was unclear, the user can also read the list manually to clarify what was heard.

3.2.6 Input

Input in a mobile environment can also benefit from multimodality. If the user is unable to use the keypad on the cell phone for some reason, audio input can be used to continue interaction with the system.

Normal input on cell phones, using the keypad or stylus, has been shown to drop in effectiveness by up to 30% when being used whilst walking (Brewster, 2002). The small keys on the keypad make it difficult to find the right key when the user is walking or moving. Moving around makes it difficult for the user to place the stylus exactly where he/she wants it. This is due to the size of the targets as well as the stylus moving around in the hand of the user.

Both modes of input require visual user concentration. It is difficult for the user to focus on his/her current task (walking, driving, etc) and to ensure that the device is being used correctly (Brewster, 2002).

Speech input in a mobile environment on cell phones cannot yet be used without extra infrastructure, as the processing power and memory requirements are just too high

(VoiceSignal, 2001). That does not mean that speech cannot be used, it just means that speech recognition must be the responsibility of an external device. For cell phones, the user could phone the system and interact via speaking as if on a normal phone call.

Gestured input has also been used as input for cell phones (Brewster, 2002). Some cell phones are coming out with built in accelerometers providing feedback about their movement and orientation (Siewiorek *et al.* 2003). The idea is that users can shake the phone to reject phone calls, or tilt it to scroll down a page. An advantage of gestured input is that it does not require any visual attention, allowing the user to focus on more important tasks. Interacting with gestured input devices is currently difficult as the infancy of current research limits the level of support each devices offers.

Non-speech audio can also be used for multimodal input, in both an active and a passive sense. The surrounding noise levels can be detected by the system and the volume of the output adjusted without any user interaction. The system could also use sharp, loud noises as confirmation (Oviatt, 2000). Any type of loud grunt or shout could be used to accept an option, reducing the chance of a recognition error when saying “yes” or “ok”.

It is possible that multimodal input not only implies allowing more than one mode of input, but means allowing any combination of modes for input. This means that once a user starts a task in a certain mode; he/she does not have to stick with that mode in order to complete the task. An example would be to allow the user to request emails received from a contact called Ryan. Since the user will probably have more than one email from Ryan, a visual list can be displayed for the user to select the appropriate email. After the email has been selected, it can be read to the user using voice. Switching modalities on-the-fly can be very powerful, but also very confusing for the user and a lot of research has been done into finding ways of providing smooth transitions between modalities (Coles *et al.* 2003).

3.2.7 Output

The small screen size of cell phones makes it difficult to create a good visual-only user interface. This does not mean that it cannot be used however, as many mobile applications rely entirely on visual output. While text can be rendered quite effectively on these small screens, the amount of text that can be shown without scrolling is limited. As a result, icons and images are sometimes used in place of text to provide output (Brewster, 2002).

The problem with using icons and images is that since the display is so small, it can be difficult to see with clarity what the image or icon represents. Earlier models of cell phones do not support images and would be left out if icons or images were used in place of text completely (Brewster, 2002).

Audio output is very effective on mobile cellular devices (Brewster, 2002). Both voice and non-voice output is used frequently on cell phones since these devices were designed for audio output.

Speech output is usually only provided on systems that require the user to phone in as it is difficult to include speech output that resides solely on the phone itself due to the lack of processing power and memory (Brewster, 2002). The benefits of speech output are that it does not require the users' visual attention and can be used independent of what the user is doing visually. If a hands-free kit is used it can also be used without direct input from the user. This allows the user to focus on driving and also reduces the errors made using normal cell phone input methods whilst walking (Brewster, 2002).

Non-speech audio outputs are also useful for multimodal mobile systems and are less intrusive. While the amount of meaning for each sound is less than that of speech audio, simple sounds can form part of a larger interface providing audio feedback for the interaction between the system and the user. There are two main presentation techniques for non-speech audio, namely Auditory icons, and Earcons (Brewster, 2002).

Auditory icons were developed by William Gaver (Gavar, 1989). They are defined as "*everyday sounds mapped to computer events by analogy with everyday sound-producing events. Auditory icons are like sound effects for computers*". The idea is to use small, short sound clips to represent real events. This works on the theory that people prefer not to listen to the volume and pitch of the sound, but rather the source of the sound (what sound it is).

Earcons were developed by Blattner, Sumikawa and Greenberg (Blattner *et al.* 1989). Earcons use abstract tones in a combination (similar to syllables) in order to create musical "words" with associated meanings. Contrary to auditory icons, there is no clear link between the earcon and its meaning. The meaning associated with each earcon must be learnt; these meanings must be combined to create a complete understanding of the system output.

The tactile sense can also be used for output with a mobile device. Many existing phones support a vibrating alert system that can be used to notify the user of incoming calls or messages (Brewster, 2002). It is difficult for mobile systems to take advantage of the vibrating alert system on the cell phone, but certain makes – such as Samsung cell phones - do allow software designers the use of the vibrating alert system.

3.3 Context Awareness

Dey *et al.* (2001) define context as "*any information the can be used to characterize the situation of an entity, where an entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.*" The definition is very broad and includes all possible aspects of what context is and how it affects the data. The definition is perhaps too broad, and can be refined to define context as the situation and environment in which a mobile device or user is. This definition is more specific to mobile devices and does not include all possible aspects but is general enough to be expanded depending on what contexts can be determined. The context of use is the general environment in which the mobile device is being used in. This general

environment may be determined by single sensors or by using multiple sensors and fusing their values.

Looking at the above definition of context, we can split the context of any mobile application into three categories (Schilit *et al.* 1994):

1. computing context;
2. user context; and
3. physical context.

The computing context contains elements such as network connectivity, available bandwidth, costs and other nearby resources. User context is information about the user, the users' name, location, or even other nearby mobile users. Physical context describes the environment of use, the temperature, altitude, lighting and noise. Each of these can be further broken down into various other elements. The physical element of light can be broken down into the light level, the wave length of the light, if the light is flickering or not, etc (Figure 3.2).

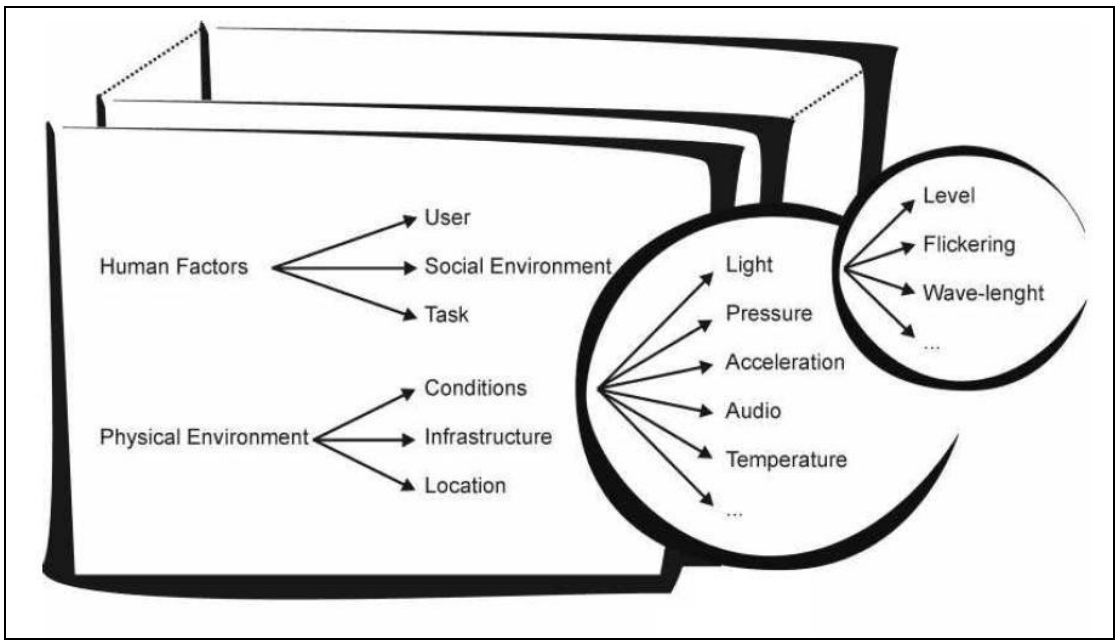

Figure 3.2: Breakdown of context elements (Schmidt *et al.* 1999)

The context of use determines which mode of communication the user would prefer to use. For instance, many people feel it is not suitable to speak on a cell phone in a public setting (Love and Perry, 2004). Deciding if the device is being used in a public

setting or not can be determined by using a variety of sensors. Certain situations imply a need for a different mode of communication rather than simply preference. In a noisy environment, if the device can detect that the ambient noise level is too high to use voice input, it can automatically switch to a visual mode of communication.

Mobile devices can benefit from the use of context awareness in several ways (Schmidt *et al.* 1999). The more a mobile device knows about its environment, the better it can support the user to perform desired tasks. This is an obvious benefit from a usability viewpoint and could help generate mass adoption of mobile applications.

3.3.1 Acquiring Context

There are two ways of acquiring the context of a mobile application (Schmidt *et al.* 1999). Similar to the way of determining location, the device itself can know its own context, the network can know the device's context, or a combination can be used. The device can sense certain contexts that the network will never be able to determine itself, and the opposite case is also true. Combining the sensory data from both the device and the network allows a more complete picture of the context to be developed. Thus the best approach to determining the context of use is to use a combination of both device known context and network known context. This implies the use of a server that has access to both types of sensors: the network's and the device's. If there are no sensors that can provide any useful information, it is acceptable to expect the user to enter the context explicitly (Schmidt *et al.* 1999).

3.3.2 Implementation Problems

The design of context aware applications has traditionally been hampered by the lack of a unified context infrastructure (Ebling *et al.* 2001). This means that a lot of time during application development is spent on creating a context service to manage, deduce and provide context to the actually application. Work is being done in the field of creating a unified context service, but mass adoption is not likely for a few years at least (Ebling *et al.* 2001).

Once the various sensors have returned the required information (light level, noise volume, etc) it is difficult to interpret this information (Ebling *et al.* 2001). It may not be possible to deduce from a flickering light level that the device is indoors. Schmidt *et al.* (1999) propose using a fusion of sensors to determine whether the user is indoors or outside.

Designers have lacked traditionally sufficient conceptual tools for the design and implementation of flexible context aware applications (Dey *et al.* 2001). This has been addressed by several authors, each suggesting their own method for modelling and representing contexts, sensors, users, etc. Dey *et al.* proposed a method using Widgets (not GUI widgets), Aggregators, Interpreters, Services and Discoverers. Their framework involves splitting context acquisition from how the context is used and how the context is presented. This follows from Java's Model-View-Controller architecture, separating how the context is determined from what it means. They also emphasise separating the context acquisition from the application. This provides a common way for applications to acquire context, meaning that several applications can use the same sensors. The reason for this is to enable multiple developers to use existing sensors in their applications without having to write scripts or objects to perform something that has already been done.

Henricksen and Indulska proposed another framework for designing context aware applications, called the Context Modelling Language (CML) (Henricksen and Indulska, 2004). The CML is based as an extension to Object-Role Modelling (ORM). The abstraction of the application from sensor data is left up to the designer and as such, the CML does not encourage code reuse between multiple designers and applications. CML forces designers to follow a more infrastructure centred design, relying on sensor technology currently available.

There are currently only a few models and guidelines that support iterative development of context aware applications (Dey *et al.* 2001). This makes it difficult to provide an extensible application that can operate independently of new sensor technology or altered functionality. An ideal framework would provide applications with access to sensor information, without having to worry about the sensors providing different types of values, the sensors changing completely, or becoming

completely replaced. The Dey *et al.* framework allows for this level of abstraction by separating the way sensor information is interpreted from the way it is used.

3.3.3 Sensor Technology

There are many different ways of sensing the environment around a device. Mobile devices are traditionally not designed to include sensors that can determine information about the environment in which they are being used, but this is changing as more research is being done on creating context aware mobile devices (Chen *et al.* 2000). There are multiple sensors that can be used to determine the context of a mobile device (Chen *et al.* 2000; Schmidt *et al.* 1999a; Schmidt *et al.* 1999b; Brown *et al.* 2000; Chen and Kotz, 2000; Siewiorek *et al.* 2003; Mitchell, 2002). Table 3.1 shows a summary of what each sensor does and the information it can provide as well as it's relevance to this research.

When determining a users' environment, a combination of the sensors in Table 3.1 should ideally yield an accurate view of the users' environment. Having all possible sensors on a mobile device is useless unless that information can be interpreted and the mobile application can adapt to the most suitable mode based on that information (Winograd, 2001).

Sensor	Description	Relevance	Type of Information
Optical	A cell phone's camera can supply visual information of the environment from which certain deductions can be made.	Light is something that can be accessed easily and simply. Knowing if the light is artificial or natural can determine the surroundings.	- Light level - Wavelength - Type of light (artificial, sunlight)
Audio	The microphone that all phones come with can be used to deduce aural clues as to the environment.	It is not currently clear whether or not the microphone on the phone can be accessed directly. The background noise in a mobile environment could prevent audio being used.	- Ambient noise level - Loudness - Type of background noise (voice, mechanical)
Motion	Using accelerometers it is possible to detect the motion and direction of a mobile device. There are only a few devices that come with built in accelerometers, but information on direction and speed can be used in conjunction with location to provide greater positioning accuracy.	Limited use as only a few phones come with accelerometers and there is little information on how to access the accelerometers readings.	- Speed - Direction
Location	Knowing a mobile device's location has many documented benefits. Certain locations can be earmarked for certain profiles ensuring the correct mode of communication in the right location. Historical data can also be mined to determine what modes users prefer in what locations.	Cell phone location can be determined using Bluetooth sensors or can be requested by the network.	- Position - Co-location of other users - Proximity of devices and services
Diary /	If the users diary is stored on the phone the device will	The diary is an easily available source of	- Scheduled appointment types (interruptible or not)

Calendar	know if the user intends to be in a meeting or otherwise occupied.	environment information and should be used if provided.	- Appointment duration
Time	Used mainly in conjunction with the diary, the time can also be used to determine the mode. Late at night text might be better as the user will probably be sleeping.	Time is easily accessible and certain obvious inferences can be made from time.	- Current Time and Date
Specialised	This is the class of sensors that detect things like the temperature, air pressure, etc. These have limited application but can be used to paint a more complete picture.	There are no cell phones currently on the market that support any of these Specialised sensors.	- Air pressure - Temperature
Biometric sensors	These are traditionally used to validate the user, but can be used to determine which other users are nearby using voice recognition or image recognition.	Not applicable as recognition usually needs neural networks.	- Pulse rate - Blood pressure - Skin moisture
User provided	The user can also explicitly supply data about the environment. This should be avoided as much as possible, but can be useful. If the system determines an incorrect environment, the user should be able to over-ride it.	Should be used to test other simulated sensors, thus simulating context data that might not be available yet. Could also allow users to force the mode they want in exceptional cases.	- Profile of phone (silent, loud) - Default preferences

Table 3.1: Summary of possible sensors and information provided

3.3.4 Interpreting Context

There are various ways in which the context of a mobile device can be used once it has been determined (Brown *et al.* 2000). An application may use context to trigger an event. Location-based m-commerce would fall into this category, where as soon as a user is determined to be near a certain area of interest, he/she is notified of any specials or products on sale near that location.

Applications can also use context to identify events. If John was in a meeting 11 months ago, with Paul and Melody, he could request minutes from that meeting as long as he specifies enough context to uniquely identify that particular meeting (in this case, specifying that it took place more 10 months ago, and Paul and Melody were present). An example of this application is the *memory prosthesis* application at Xerox European Research Centre in Cambridge (Lamming *et al.* 1994).

Reminders can also be facilitated by context awareness. Reminding a user of something the next time the user is in the vicinity of another user can use location awareness to determine when the two users are close enough for the reminder to trigger (Brown *et al.* 2000).

This research focuses specifically on using context to streamline interaction. Context awareness can be used to facilitate any process that could otherwise cause an interruption in the flow of a situation (Abowd and Dey, 1999). In the case of incoming email data, determining which mode is most suitable for the user's current context could streamline the interaction between the email data and the user.

Dey *et al.* not only propose that sensor information should be abstracted from the application, but that context should be interpreted by an external component (Dey *et al.* 2001). The purpose of this is so that multiple applications can use the same determined context without having to infer the same result as the other applications. Obviously, not all applications will be interested in all the contexts. If two or more applications are interested in whether or not a user is in a meeting, having one component that determines this by using multiple sensors makes it possible for both

programs to be designed and implemented faster than if the programmers had to provide the interpretation from the sensors themselves.

3.3.5 Sensor Fusion

Context information derived solely from single sensors has limited applicability (Schmidt *et al.* 1999b). Being able to determine if the environment is loud (as opposed to quiet) can only be useful if the application's response is based solely on the surrounding noise level, but this is rarely the case (Chen and Kotz, 2000). Most context aware applications take information from multiple sensors and use all the information to form a representation of the user's environment.

Dey *et al.* have provided an easy to understand, flexible and adaptable solution to combining various sensor information (Dey *et al.* 2001). Each context sensor runs independently of the application, as is the case for the components that synthesise the information from the various sensors. These synthesisers are called "Aggregators". These aggregators are accessed by an application in the same way that a normal sensor is accessed. Since each aggregator runs independently of an application, it can be used at the same time by more than one application. The only task aggregators perform, is to take the input from multiple sensors and apply a set of rules on the input which provides an output that reflects the environment. Taking the users' location and surrounding light level as well as the ambient noise level, could be used to determine if the user is in a meeting or not. This information is of a higher level of abstraction than the individual light, noise and location values. Creating chains of abstraction is also possible, where the meeting aggregator could be combined with the users' diary information to determine if the user is in an interruptible meeting or not.

Taking information from various sensors can also be used to prevent errors in determining the user's context. The Technology for Enabled Awareness (TEA) assigns a certainty to each context which represents the certainty that the user is currently in that context (Schmidt *et al.* 1999a). Another approach (and substantially easier to implement) that has been used is to assign a priority list to the various sensors (Siewiorek *et al.* 2003). If sensor A determines a certain context, but it

contradicts with what sensor B determined, the one with the higher priority overrides the other.

3.4 Requirements

There are certain requirements that must be satisfied before multimodal context aware interaction can take place in a mobile environment. These requirements are based on the fundamental differences between interaction with a standard desktop and multimodal context aware interaction in a mobile environment.

Desktop systems are used in the same environment while mobile systems are designed to be used in multiple environments and situations. Mobile systems have to allow for these changing situations and environments without the user necessarily specifying that a change in environment or situation has occurred. Multimodal systems for mobile devices need to allow switching of modes based on the context of use. Determining the context of use is often a complex task, as sensors are rarely accessed in the same manner, from the same device. This often leads to context aware systems having complex distributed designs.

While desktop systems may use multimodality as a tool to simply enhance the user interface (Bolt, 1980), mobile systems use multimodality to replace any one of the modes for complete interaction of the system. This means the entire system has to be controllable from each supported mode without any other modes being needed. This is to provide functionality for the times when the user is completely unable to use a certain mode.

Designers of multimodal context aware applications suffer from a lack of abstraction to context. There are very few infrastructures or models that specify how information determined about the environment can be abstracted, and how to choose the correct mode of output based the abstracted information (Dey *et al.* 2001). Standard software engineering practices, such as UML modelling, are currently insufficient for designing multimodal context aware applications. The lack of abstraction also makes it difficult for applications to be designed to support new sensor technology without having to rewrite parts of the (or even the complete) application. Abstracting how

context is determined from how it is used has been shown to be useful is resolving these issues (Winograd, 2001).

As was described in section 3.3.1, context can be acquired from the mobile device, from the mobile device's network, and a combination of both. In contrast, a standard desktop environment's usual input devices (the keyboard and mouse) are directly attached to the PC. It is a simple task for an application to check the status of these input devices. For multimodal context aware mobile applications, the application needs access to sensors that may be accessible from the device or from the network only (Dey *et al.* 2001). This causes difficulties when a proper architecture for distributed communication has not been defined. The problem is compounded when multiple applications that use the same sensors are considered. It is not reasonable to assume that all the applications that use a particular sensor will be directly connected to the device on when the sensor is located. It is necessary for there to be a way for applications to link with the various sensors and for sensors to link with each other in such a way as to appear transparent and seamless to the users and designers.

The history associated with context can be used to provide valuable information, such as movement, but very few architectures provide the facility for context history. In most cases, history of context is left up to the application to manage (Mitchell, 2002). The lack of infrastructure for managing context history has made designing context aware applications more difficult than if the sensors supported history by default. If the sensors that provide the context information are located on multiple devices, it is also possible that the information they provide will not be synchronised. There needs to be a way to synchronise information received from various sensors.

These difficulties mean that for a mobile service to support multimodality and context awareness, it has to meet certain requirements (Table 3.2).

	Factor	Requirement
1	**Abstraction**	Abstract context acquisition from context use.
2	**Flexibility**	Support new sensor technology as it becomes available.
3	**History**	Support context history.
4	**Distributed**	Support distributed communication.
5	**Time**	Support time synchronisation between multiple sensors.
6	**Mode switching**	Allow for dynamic switching of modes.
7	**Mode functionality**	Provide complete functionality with each mode independent of other modes.

Table 3.2: Requirements for multimodal mobile systems

If we include the requirements of a mobile notification service outlined in Chapter 2 (Table 2.2), we can create a list of requirements for a mobile notification service using multimodality (Table 3.3).

		Factor	Requirement
Mobile Notification	1	**Mobility**	Take the mobility of users into account.
	2	**Content formats**	Process multiple content and interface formats.
	3	**Scalability**	Support large amounts of subscribers and publishers.
	4	**Security**	Support security and privacy of information.
Multimodality and Context Awareness	5	**Abstraction**	Abstract context acquisition from context use.
	6	**Flexibility**	Support new sensor technology as it becomes available.
	7	**History**	Support context history.
	8	**Distributed**	Support distributed communication.
	9	**Time**	Support time synchronisation between multiple sensors.
	10	**Mode Switching**	Allow for dynamic switching of modes.
	11	**Mode functionality**	Provide complete functionality with each mode independent of other modes.

Table 3.3: Requirements for a mobile notification service using multimodality

Using these requirements, we can evaluate and compare models, frameworks and architectures for mobile notification using multimodality and context awareness. After comparing the models, an appropriate one can be selected for use with email data.

3.5 Models

A number of models exist for mobile notification services using multimodality and context awareness. Not all of the models are designed for a cell phone interface, and many of these models cannot be applied to an email data application domain. Comparing these models using the requirements defined in section 3.4 will allow the selection of the model that is most suitable of being used to create a mobile notification service of email data using context awareness and multimodality.

3.5.1 W3C Multimodal Interaction Framework

The W3C Multimodal Interaction Framework (Larson, 2003) identifies mark-up languages for use with multimodal systems. The framework also identifies the major components that multimodal systems should possess and how they relate to each other. The framework includes many input and output modes currently in use and can also be extended to include additional modes as they become available.

The W3C framework provides most of the foundation needed for multimodal interfaces to become feasible. The purpose was to create a standard for multimodal applications, similar to the W3C Document Object Model (DOM). The result is a framework which has been used as a basis for many multimodal applications and models. The framework was not designed specifically for a mobile environment, but is not restricted from being applied to a mobile environment.

The framework consists of various components and explains how they interact as a whole. These components are a human user, input, output, an Interaction Manager, a session controller and a system and environment component (Figure 3.2).

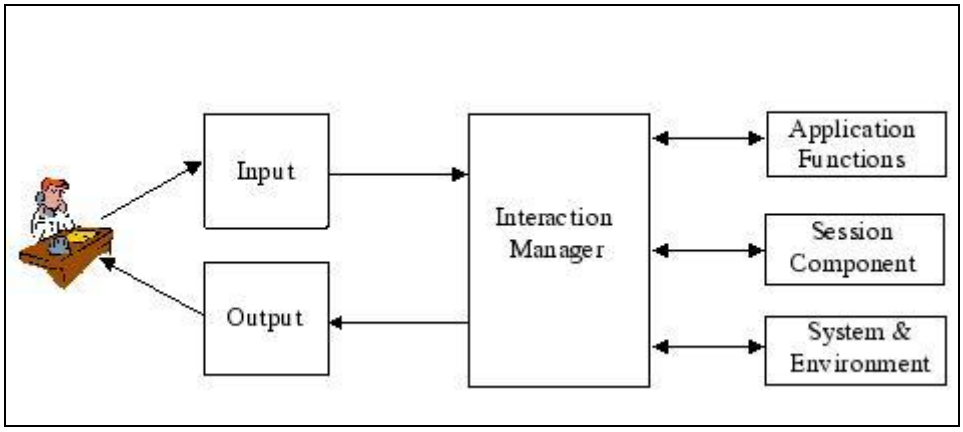

Figure 3.3: The W3C Multimodal Interaction Framework (Larson, 2003)

The input is provided by multiple input modes, speech, audio, keyboards and other modes (Figure 3.3). The input from each device is captured and translated by a Recognition component into a form for later processing. The Recognition component uses a grammar that defines the way the modes interact with each other. The results of the Recognition component are then interpreted. Each input mode has its own Interpretation component which identifies the meaning of the input. This allows for multiple phrases to have the same meaning without providing for those meanings explicitly. The output from the Interpretation components is then combined by an Integration component. Other data not explicitly from user input can also be included in this step. The processed input is then passed to the Interaction Manager. The Interaction Manager also accepts input from a System & Environment component. This component is responsible for collecting and changing contextual information about the environment. The framework is designed to combine input and output with respect to context, for example a component that determines the light level in a room, would be the same component that turns that room's light on or off.

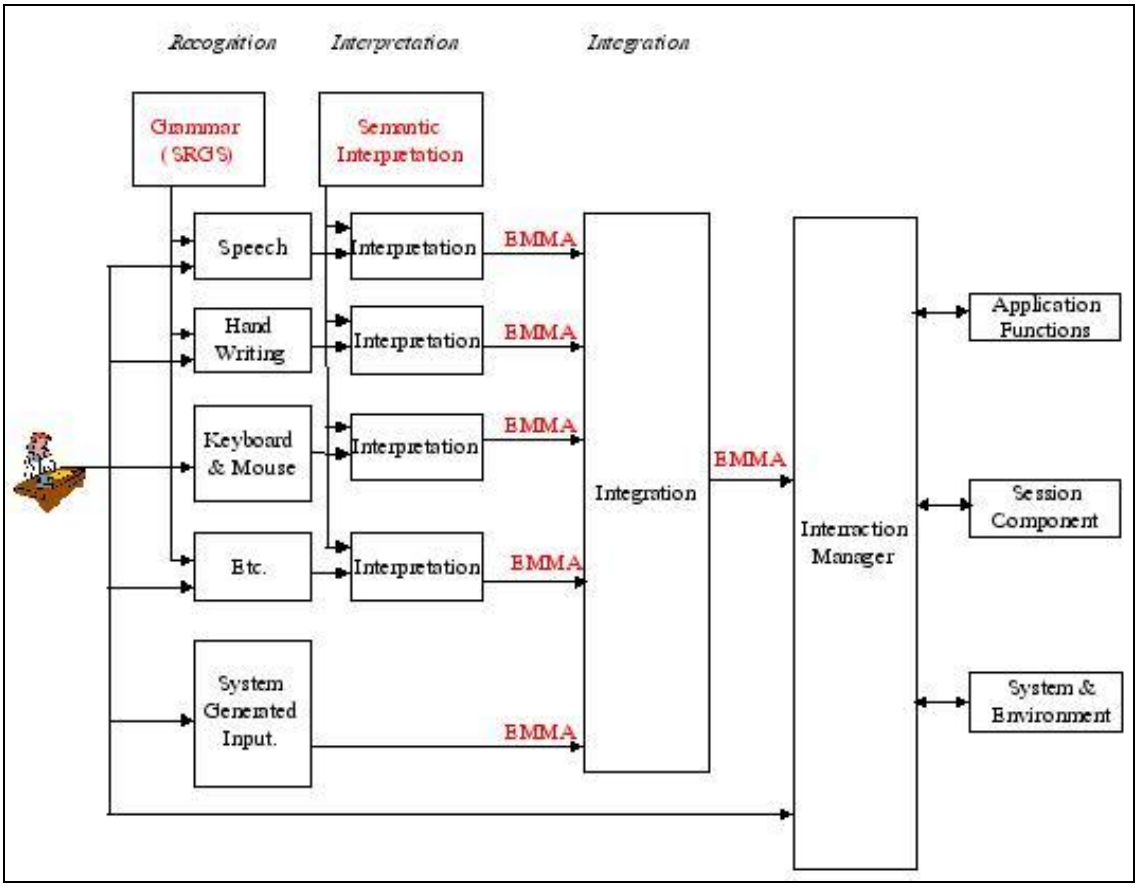

Figure 3.4: W3C input components (Larson, 2003)

The Interaction Manager component does not have an exact specification. It is a logical component dependent on the application. The Interaction Manager component accepts a string as input. This means that the Interaction Manager component has no idea about what devices or modes were used for the input, all it knows is that it has received an input of certain data. The Interaction Manager component can receive other forms of input from the Session Controller component or the system and Environment component. After it has finished computing based on the set of input, it produces a string as output. No modalities are specified for output; all that is given is certain data that must be output by the Output component.

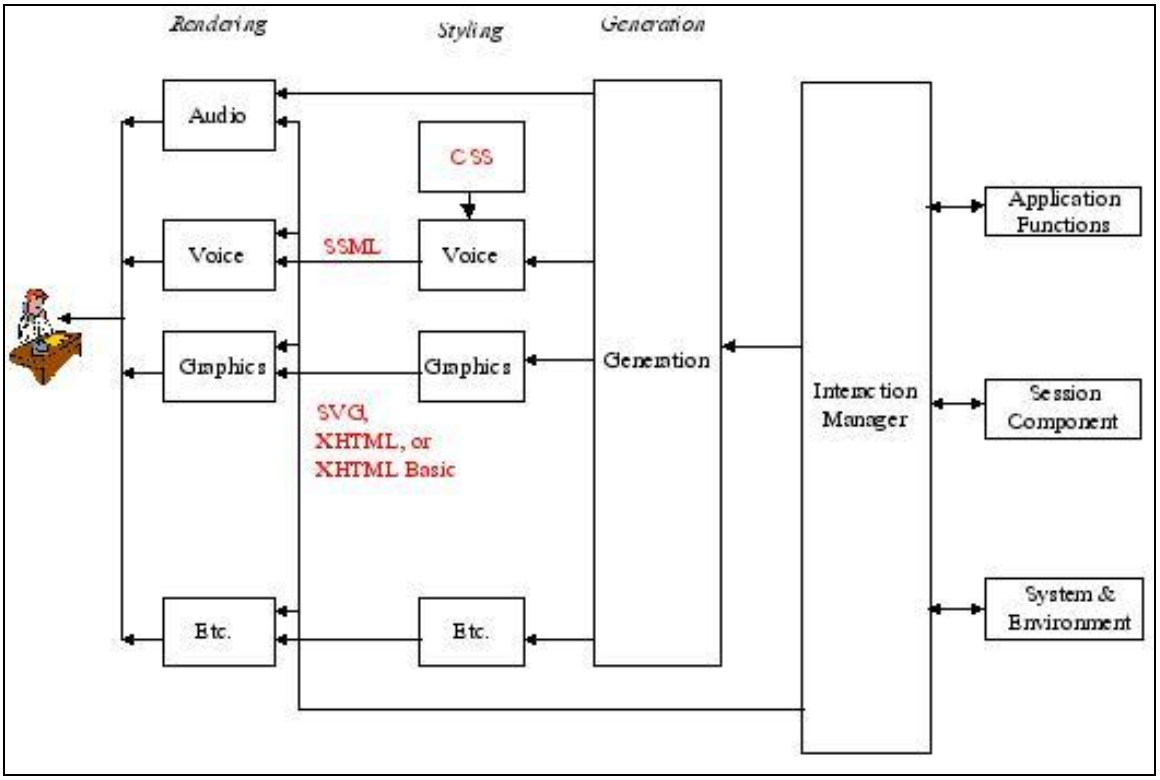

Figure 3.5: W3C output components (Larson, 2003)

The Output component (Figure 3.4) accepts as input a string from the Interaction Manager. This output is processed by the Generation component. The Generation component determines which modes will be used for output. The Generation component may select a single mode, or multiple modes to represent the same data. The data can also be broken into parts and each part represented on a different output. It is possible for the Interaction Manager to bypass the Generation component in the case where the output has to be of a specific format. The Styling components decide and add information about how the output is to be presented. The Rendering components take the information and convert it into the correct format and output the result to the user. Each Styling and Rendering component is implemented based on the mode it is going to use.

The framework does not provide anything more than a general overview of how the components of models for multimodal systems operate and interact with each other. The framework also does not specifically discuss mobility, but the mobility can be included as two extra components, one between the Input component and the Interaction component and the second between the Output component and the

Interaction component. Distributed communication is also not implicitly provided for and the designer has to design mechanisms for receiving data from distributed sensors.

The amount of subscribers and publishers that can be handled is dependent on the Interaction Manager. If the Interaction Manager for a particular model is defined to allow for flexibility, then the amount of publishers and subscribers is not a problem. The framework meets the following requirements (Table 3.3):

- it supports the mobility of the users;
- it processes multiple content and interface formats;
- context acquisition is abstracted from context use;
- distributed communication is catered for;
- provides complete functionality for each mode;
- context acquisition is separated from context use; and
- allows for dynamic switching of modes.

The framework does not, however, support any security or privacy and as mentioned above, the number of subscribers and publishers is dependent on the Interaction Manager. There is no support for new context sensors that could be developed unless the designer specifically caters for this, or a separate component that performs this task is included. Context history, distributed communication and time synchronisation are also not supported by the framework, leaving the designer to solve these problems. The W3C framework does not support the following requirements (Table 3.3):

- no context history is supported;
- time synchronisation between multiple sensors is not supported;
- the number of subscribers and publishers is limited to the Interaction Manager;
- no security or privacy is provided; and
- the context awareness is designed around particular devices and for new devices to be supported; the application has to be changed.

3.5.2 Nomadic Radio

The Nomadic Radio project (Sawhney *et al.* 2000) was an investigation into wearable computing. The system allows the user to stay in touch with services like email, voicemail, news broadcasts and calendar events. Messages were filtered and sorted, then placed into categories. Users can then select a category and browse, save or delete messages in that category. Textual messages are read to the user using a synthetic voice while audio messages are played to the user using spatial audio streams. Navigation is performed by simple voice commands issued to the system or by a set of buttons for situations where aural navigation is not suitable.

The hands-free wearable device that is used is a modified version of the *SoundBeam Neckset*. It consists of two directional speakers located on either side of the users head, and a directional microphone that rests on the users' chest (Figure 3.6). The volume of the directional speakers can be adjusted to a level where only the user can hear them without intruding on other people.

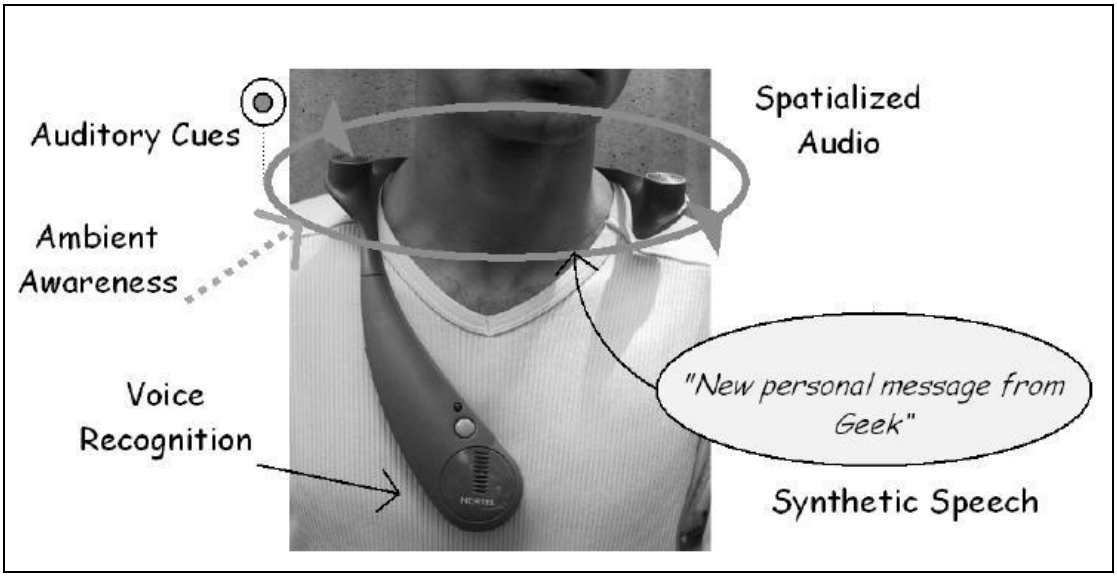

Figure 3.6: Nomadic Radio wearable audio device (Sawhney *et al.* 2000)

Nomadic Radio was designed using client/server architecture. The client is a Java-based program running on the wearable PC. The server is a set of processes running on a Sun SPARC station. Each process listens to a particular media source, email, voicemail, news, calendar, etc. When a media source is updated, the client is notified which then alerts the user using spatialised audio. The volume of the output is based

on the surrounding noise level and whether the user is currently engaged in a conversation or not.

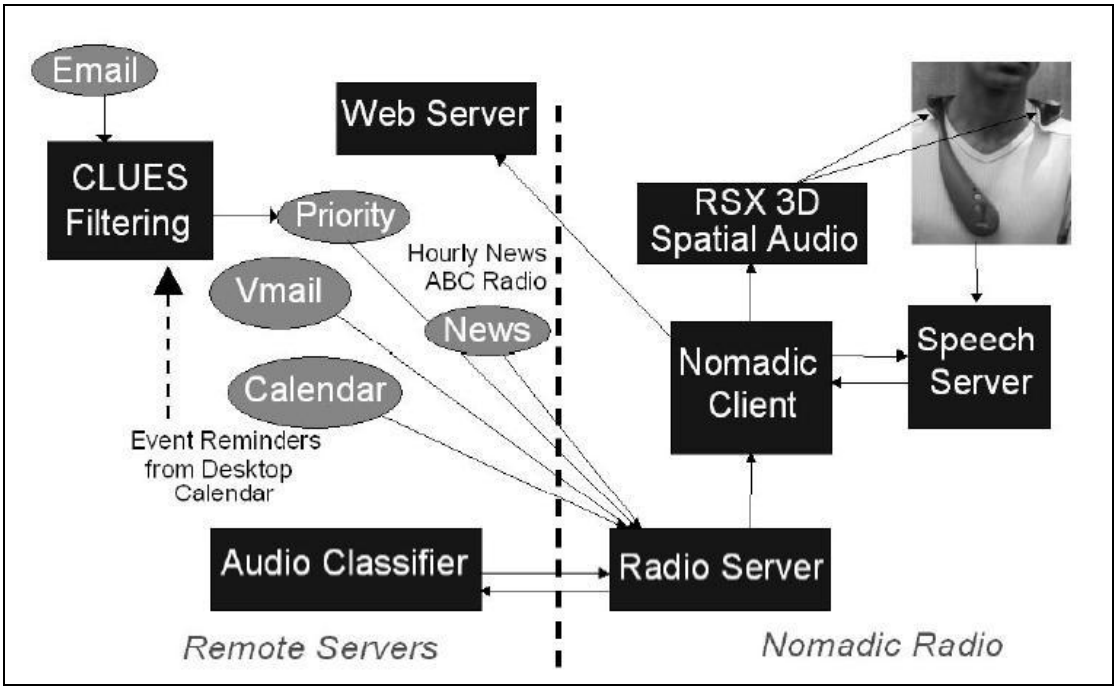

Figure 3.7: Nomadic Radio system architecture (Sawhney *et al.* 2000)

Spatialised audio is a technique where messages and audio are played in particular spatial locations. Using spatial audio makes it easier for users to get an idea of what source the message is coming from without having to listen to the entire message.

In Nomadic Radio, voicemail and news arrive at different times so playing their messages at the approximate time of their arrival makes it easy to distinguish the two. Messages are played in chronological order around the users head. Users can then obtain an approximate time of message arrival from the message's position whilst being played.

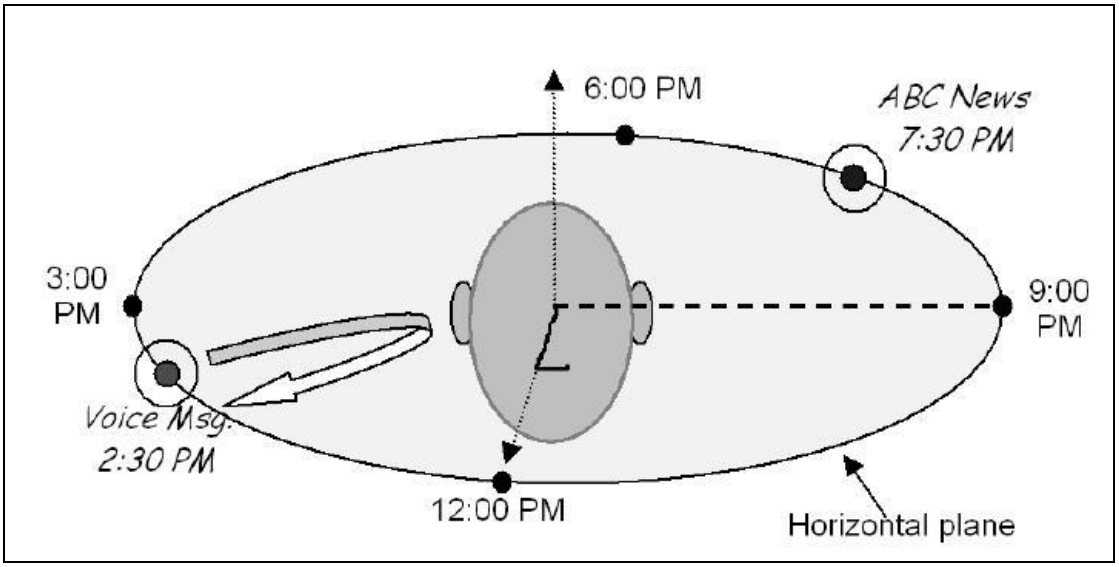

Figure 3.8: Nomadic Radio spacialized audio (Sawhney *et al.* 2000)

Users can also scan messages in Nomadic Radio. By issuing a "scan messages" command, each message is played for a short duration; going in reverse order from the latest message to the oldest message. After each message has been played, there is a two second delay where the user can select the last message played. Issuing any command during scanning will stop the scan. Spatial scanning is also used for messages to allow even faster scanning of all messages. Each message cycles through the user's centre listening space and fading off to the side while the next message is cycled into the user's centre listening space (Figure 3.8). This allows a small part of the messages to overlap, but the user can still listen to the previous message by shifting his/her listening centre. This auditory scanning allows users to listen to all messages faster than with just normal scanning methods.

The architecture of Nomadic Radio is modular and extensible with respect to the services that each user can subscribe to. There is however a limitation on the number of subscribers that can use the system. Each subscriber launches several processes on the server and with thousands of users; the load on the server would be too much. Since it runs on an 802.11 network, the range of the device is suited only to local areas (inside an office situation). Nomadic Radio satisfies the following requirements for a mobile notification service including multimodality and context awareness (Table 3.3):

- it takes the mobility of the user into account within a local area;
- processes multiple content formats;
- supports large amounts of publishers (not subscribers though);
- provides support for distributed communication and time synchronisation;
- each mode can operate independent of the other; and
- modes can be switched dynamically.

The Nomadic Radio architecture does fail to meet certain requirements however, namely (Table 3.3):

- does not support a large number of subscribers;
- context abstraction and history is not supported;
- does not support new sensor technology as it was designed around a particular device;
- provides no security or privacy; and
- the mobility of users is limited to local 802.11 ranges.

3.5.3 IBM Intelligent Notification Service

The IBM Intelligent Notification Service (IBM INS) is a model for context aware multimodal notification of events occurring on the web (Bazinette *et al.* 2001). The model allows users to specify events in which they are interested. The web is then monitored for these events and the user is notified once these events occur. Notification takes place using whichever device and mode is most convenient for the user. The convenience is determined by using context awareness and user preferences in order to detect the user's environment and mode best suited for that environment.

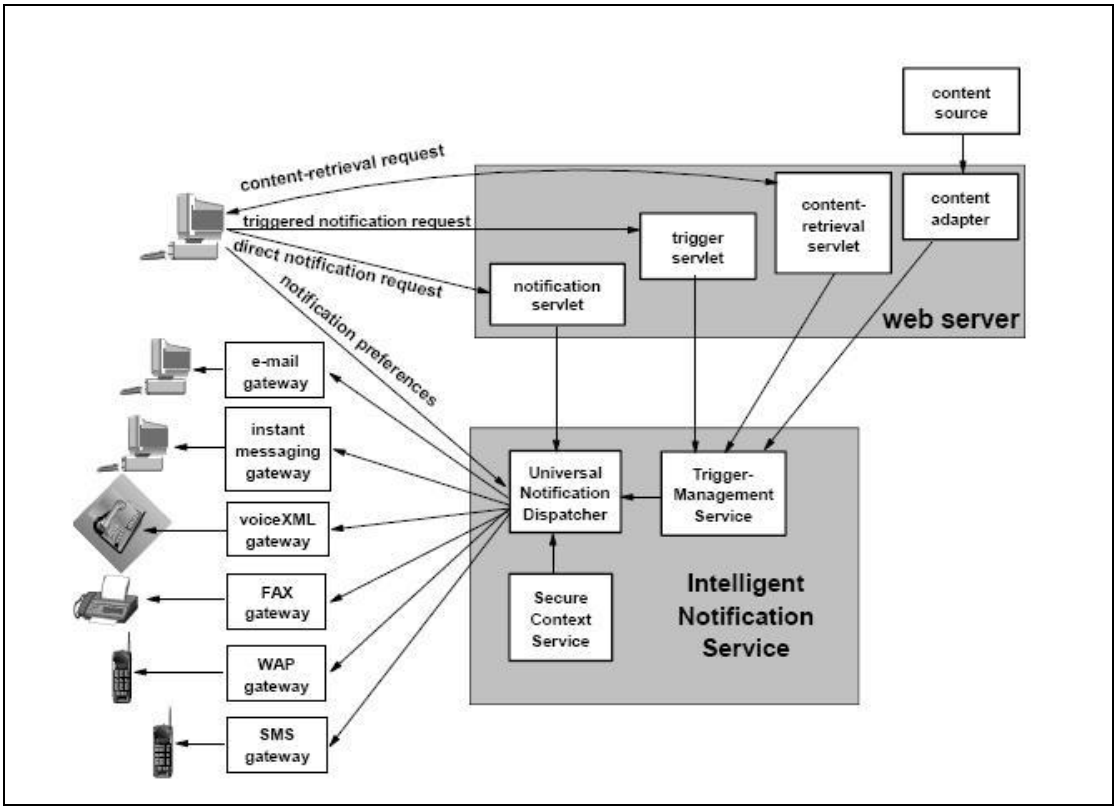

Figure 3.9: Architecture of the Intelligent Notification Service (Bazinette *et al.* 2001)

The IBM INS consists of three major components, the Trigger Management Service, the Universal Notification Dispatcher and the Secure Context Service. The Trigger Management Service is responsible for checking for events which subscribers wish to be notified of, the Universal Notification Dispatcher notifies the subscriber that the event has occurred along with any information that the subscriber requires about the event. The Secure Context Service is aware of the subscriber's context and informs the Universal Notification Dispatcher what information may affect the type mode used for notification. These three components are explained in greater detail in the following sections.

The Trigger Management Service (Figure 3.10) contains a list of subscriber's events of interest and listens on the web for the occurrence of such an event. If an event occurs, information associated with the event may be stored in the Content Store. The purpose of this is to allow subscribers to access any additional information about the event at a later stage. Each trigger is associated with a particular event. When an event occurs, the trigger's call-back method is invoked. The call-back method could notify the system that the trigger must be deleted from the Trigger Manager. This is

to cater for cases where a subscriber may only be interested in the first occurrence of an event. The Trigger Management Service could also request notification to the subscriber. Note that an occurrence of an event does not automatically mean that a subscriber is notified.

The Trigger Management Service contains two data stores, one that stores information about the registered triggers, and one that stores information about the events that have occurred. The Trigger Store is used to load all the registered triggers as soon as the application starts. This helps with system recovery if a failure occurs. The other data store is used to store information relevant to events that occur. If additional information about an event is required to be stored, the subscriber can access it at a later time from this data store. A scavenger thread is run periodically to remove archaic data from this data store.

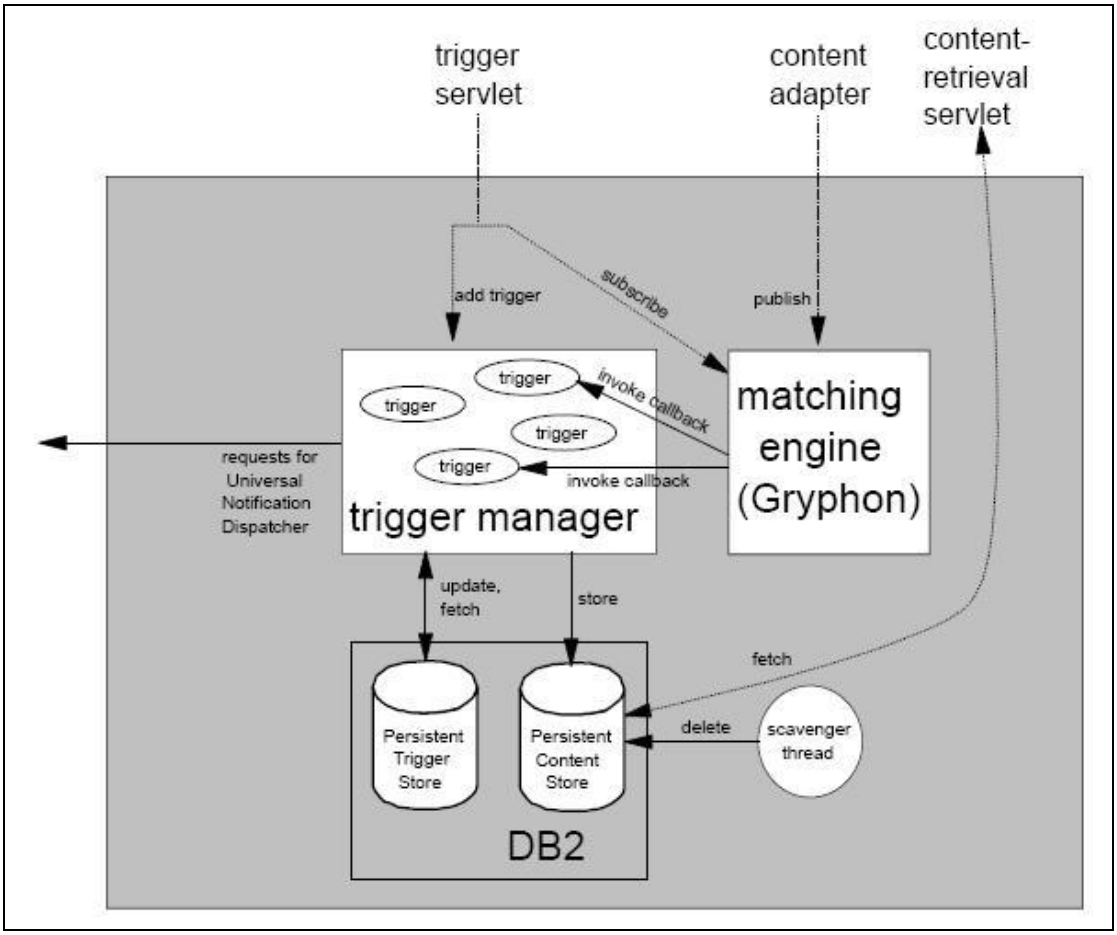

Figure 3.10: IMB INS Trigger Management Service (Bazinette *et al.* 2001)

The Universal Notification Dispatcher (Figure 3.11) sends the messages to the subscribers based on their preferences. Each subscriber is identified by a unique value, regardless of the device the subscriber uses. The Universal Notification Dispatcher has a request queue, which contains requests for notification. Notification requests can be made by the Trigger Management Service as described above, or by external authorized clients. The Universal Notification Dispatcher assumes that all clients and subscribers are already validated in the system and provides no security or authentication.

Subscribers are able to specify notification preferences during subscription and are allowed to change these preferences at a later stage using a web portal. The subscriber's preferences are stored in the Preference Engine. When a notification is to be sent, the Preference Engine is consulted to determine the correct gateway to use for notification. The Preference Engine is given information about the notification to be sent, as well as the context information which is retrieved from the Secure Context Service.

Clients that send messages have no access to the Secure Context Service and are thus not privy to any information regarding a subscriber's context. The client also has no influence on the output gateway used. The chosen gateway is entirely dependent on the client's preferences and the message to be sent. Once a gateway is chosen, the Universal Notification Dispatcher formats the message (if necessary) and sends it to the selected device. Each gateway is responsible for message to a specific type of device (voice to cell phones, sms's, fax machines, etc). New devices are supported by implementing the required gateway for that device.

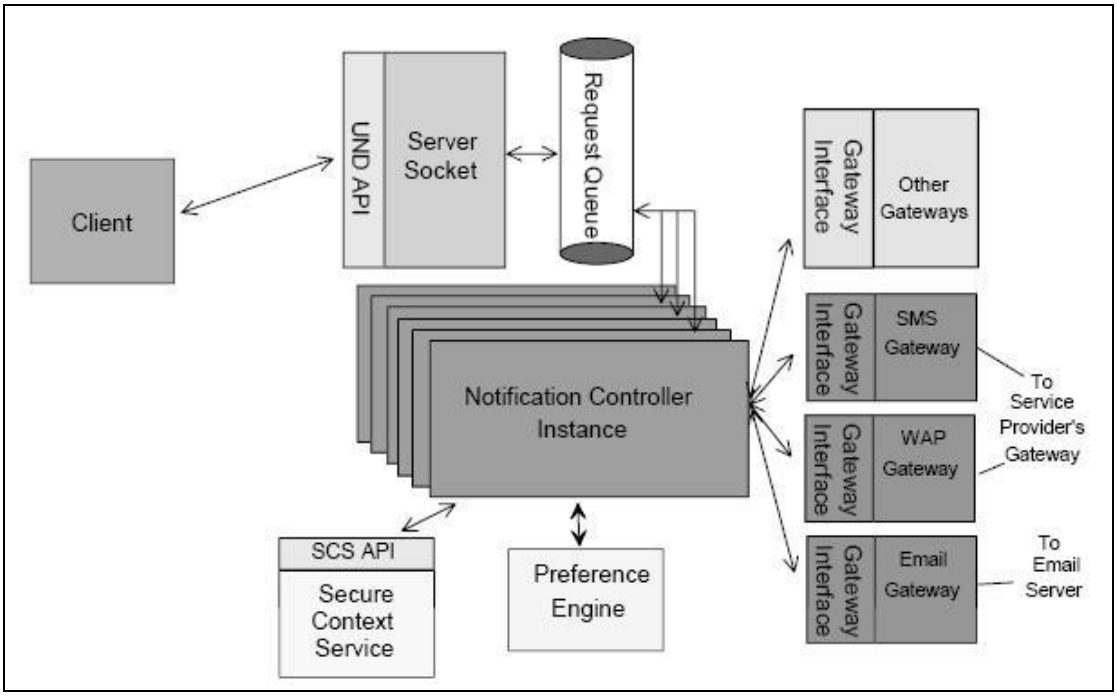

Figure 3.11: IBM INS Universal Notification Dispatcher (Bazinette *et al.* 2001)

The Secure Context Service (Figure 3.12) consists of a mediator, Context Drivers and utilities to be used internally. Applications interact with the Secure Context Service by means of an Application Programming Interface (API). When an application requests information about a subscriber's context, it sends through data informing the Secure Context Service in which contexts it is interested and the quality of the required information. The application receives this request, queries each of the applicable Context Drivers and returns the result to the application. Applications may request once-off context information, or may subscribe to certain context events either once or repeatedly.

The utility components are used to optimize the Secure Context Service. The Context Cache stores information about recently accessed context. This is in case the information is going to be accessed again and is not expected to change. It is not used to store history information about contexts. The Connection Manager is used to keep connections for the Context Drivers that are required to be open. This helps performance by minimizing the cost of re-establishing connections. The privacy engine authenticates all requests for context information based on the policies set in place by the subscribers. The Event Engine matches context events with applications

that have subscribed to those events. If a context event occurs, the Event Engine lets the appropriate application know of its occurrence.

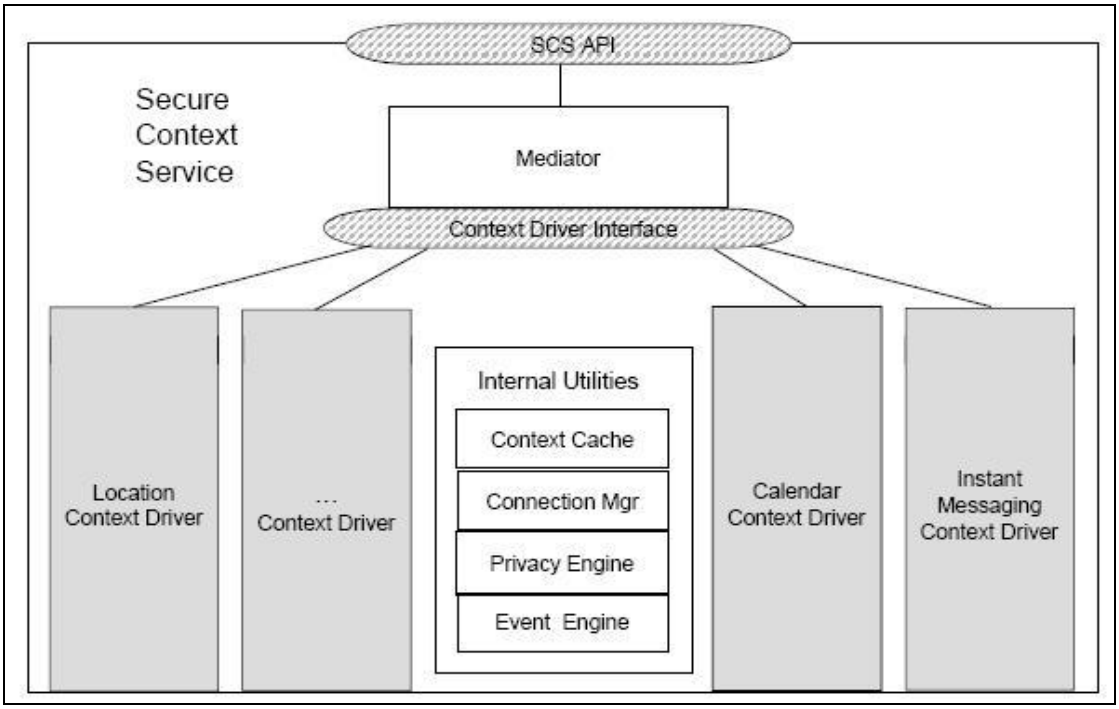

Figure 3.12: IBM INS Secure Context Service (Bazinette, 2001)

The IBM INS provides a secure, adaptable means of notification for multiple clients. It allows for distributed communication and notification across multiple modes. The following requirements are satisfied by the IBM INS (Table 3.3):

- user mobility is provided for;
- multiple content and interface formats are supported;
- a large amount of subscribers is supported and events are triggered on distributed systems allowing for large amounts of publishers too;
- context security and privacy is supported;
- new sensor technology is catered for by creating the required Context Drivers;
- context acquisition is abstracted from context use;
- distributed communication is provided for; and
- the mode of notification is dynamic.

The IBM INS is not without its shortcomings however, as it fails to meet the following requirements (Table 3.3):

- no context history is supported;
- user interaction is solely through a web interface; and
- time synchronisation between multiple sensors is not supported.

3.6 Comparison of existing models

Comparing the above models will allow us to select the most appropriate model for mobile email data notification with context awareness. Table 3.4 shows each of the models above as well as the requirements they satisfy.

	Factor	W3C	Nomadic Radio	IBM INS
1	**Mobility**	✓	✓	✓
2	**Content formats**	✓	✓	✓
3	**Scalability**	✗	✓	✓
4	**Security**	✗	✗	✓
5	**Abstraction**	✓	✗	✓
6	**Flexibility**	✗	✗	✓
7	**History**	✗	✗	✗
8	**Distributed**	✗	✓	✓
9	**Time**	✗	✓	✗
10	**Mode Switching**	✓	✓	✓
11	**Mode functionality**	✓	✓	✗

Table 3.4: Comparison of existing models

None of the models satisfy all of the requirements for a mobile notification service that uses multimodality and context awareness. In order to implement a prototype system, one of these models will be adapted to satisfy all of the requirements. Of the

models reviewed, the IBM INS model satisfies the most requirements and is well suited to the mobile email notification domain.

3.7 Conclusions

A multimodal mobile email notification system was defined as a mobile email notification system which uses multiple sequential modalities to notify the user of any received emails (Section 3.2). Multimodal systems were shown to provide higher levels of user satisfaction and provide mutual disambiguation (Section 3.2.1).

Context awareness uses information about the user's environment to alter the presentation of the data. Context awareness collects information from passive input sources (Section 3.2.2). Although audio was shown to be an effective mode of communication, it was also shown to have several shortcomings (Section 3.2.2).

The contexts applicable to a mobile application were categorised into computing context, user context, and physical context (Section 3.3). It was shown that the context of a mobile device can be determined by the use of multiple sensors. It was also shown that once the context of a mobile device has been determined, there are several ways of using this contextual information (Section 3.3.1).

The requirements a mobile notification service must satisfy in order to support multimodality and context awareness were identified. These requirements are an abstraction between context use and acquisition, sensor flexibility, contextual history, distributed communication, distributed time synchronisation, dynamic modality switching and complete functionality in every mode. These requirements can be used to evaluate and compare frameworks and models for multimodal mobile notification services (Section 3.4).

The IBM INS model was identified as supporting most of the requirements for a multimodal mobile notification system (Section 3.6). Chapter 4 will focus on the modification of the IBM INS model and the design of any additional components or modules that will be needed.

Chapter 4 Proposed Model

4.1 Introduction

This chapter proposes a model for mobile notification services that uses context awareness and multimodality to adapt to the environment (Research question 4, Table 1.1). The three tier model is an architecture for designing an application, where the application is divided into three distinct sections, namely a business logic (application) section, a data section and an interface section. This model has been shown to be very successful in creating extensible and flexible applications (Steiert, 1998). A similar approach was used in the design of the proposed model. The model design is divided into three sections, namely functional design, data design and user interface design. The functional design describes the tasks that the model supports; the data design describes the relationships between the data elements of the model and shows how these are modelled; and the user interface design details the user interaction that applications developed using the model support. This chapter discusses the architecture of the proposed model for adaptive mobile notification services followed by the functional, data and interface design.

4.2 Architecture

Based on the requirements for an adaptive mobile notification service identified in Section 3.4, this section incorporates those requirements in a model for adaptive mobile notification services. In Section 3.6, the IBM Intelligent Notification Service was chosen as the most suitable model for adaptation. This model supports most of the requirements for an adaptive mobile notification service except for context history, time synchronisation and mode functionality. User interaction in the IBM model is very limited as it focuses more on notification. The proposed model is an adaptation of the IBM model that includes support for the requirements not supported by the IBM model. The data flow in the proposed model was also changed to support a more modular design.

The model comprises four core modules and a set of content sources. Each of the modules performs a particular function while the content sources are responsible for determining when new content arrives and parsing that content into a suitable form for further use. This section identifies each module and describes its interaction with other modules. The four modules are the Trigger Management Service, the Input Module, the Context Module and the Output Module. Figure 4.1 shows the components of the proposed model and the interaction between the different modules.

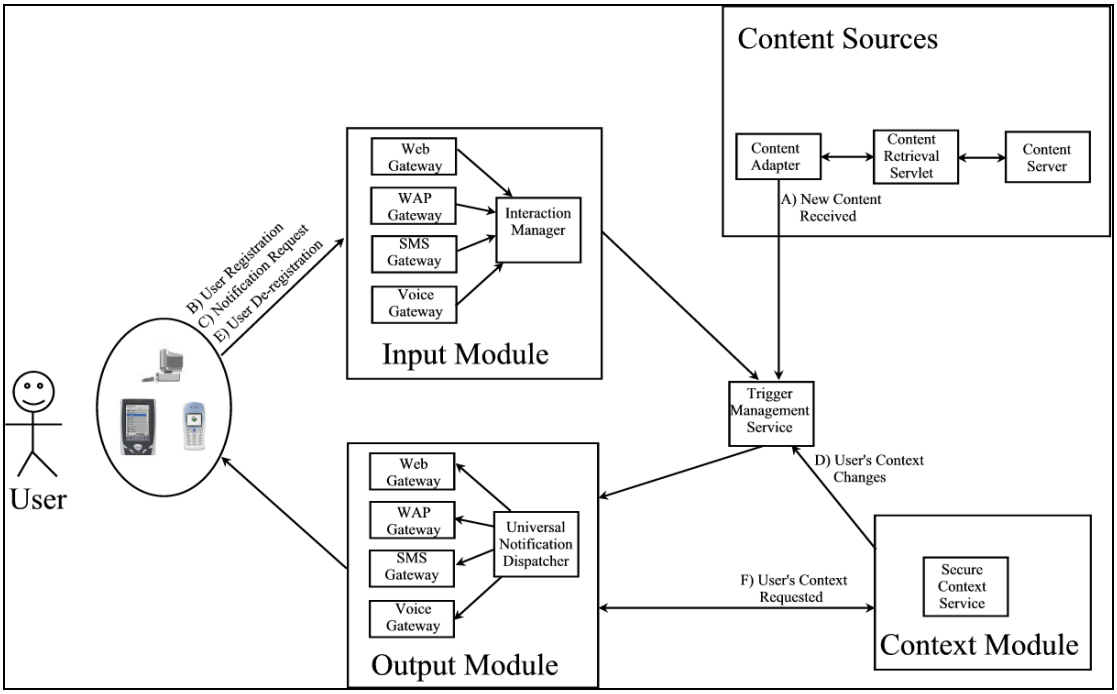

Figure 4.1: Proposed model for adaptive multimodal mobile notification service

The Input Module (Figure 4.2) is a new module not included in the original IBM model. This module is responsible for taking input from the user (regardless of mode or device) and converting it into a standard format which is device and mode independent. The IBM model allowed for only basic user input which had to be completed in the mode and on the device it was initiated. The Input Module allows for user input of one operation to span multiple devices and modes. This is achieved by the use of formal grammars. Each device passes its input to a recognition component which translates the input into a form for further processing. Once the input has been processed, it is passed to an Interpreter Component which has the responsibility of converting the processed input into a string designed to specify user interaction.

A formal grammar (or more correctly, a phrase-structure grammar) consists of a finite set of non-terminal symbols, a finite set of terminal symbols, a finite set of production rules and a starting symbol. A particular string, S, is a word from the language of a grammar if S can be produced by some combination of application of the production rules of the grammar (du Toit and van der Walt, 2002).

Each input string is then passed to an Integrator component which captures all the inputs from the different interpreters and combines them into one string. This string is used as the input for the particular operation.

The Input Module allows for user input for each operation to span various modes and various devices. The Integrator is responsible for combining all the input from the various devices and modes into one operation. For example, the user could choose to delete an email in one mode, and select the email to be deleted in another mode. The Integrator will recognise that the two commands from each mode are linked and combine them into a single operation independent of mode. Once the Integrator has combined all the inputs, it passes the result to the Trigger Management Service.

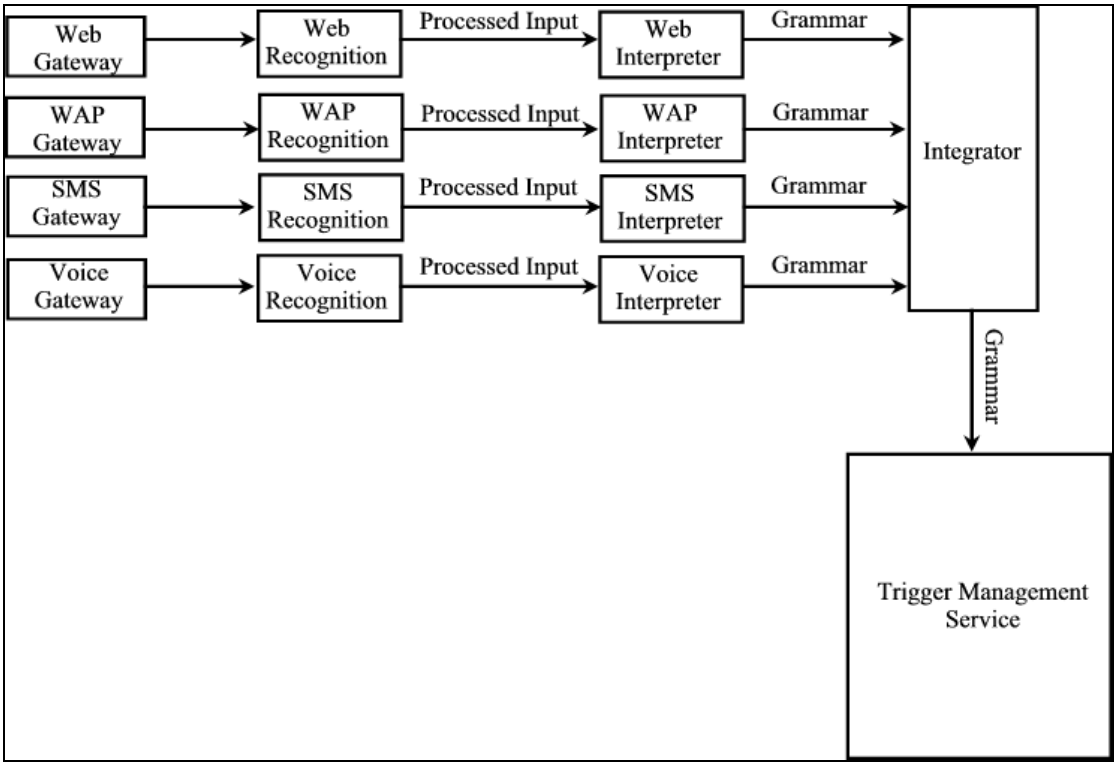

Figure 4.2: Input Module

User input can comprise user registration, de-registration, a request for a notification or a response to a notification.

The Content Sources (Figure 4.1) are the various sources of content that are notified to the users, such as e-mail servers, news web sites or stock market sites. Each content source has a Content Retrieval Servlet which is responsible for monitoring the server for new information. Note that even though the notification is push-based, the Content Retrieval Servlet could receive its information from either push or pull methods. This means that a News Retrieval Servlet could constantly poll a news web site if there is no push notification available. Once new content is available, the Content Retrieval Servlet passes the content to the Content Adapter. Each content source has its own Content Adapter in order to cater for the various types of content formats that could be available. Once the Content Adapter has converted the content into its own format, it passes the new content to the Trigger Management Service.

The Secure Context Service (Figure 4.3) is responsible for determining each user's context and passing that information to the Output Module. The Secure Context Service consists of various widgets, history stores and a Context Request Mediator.

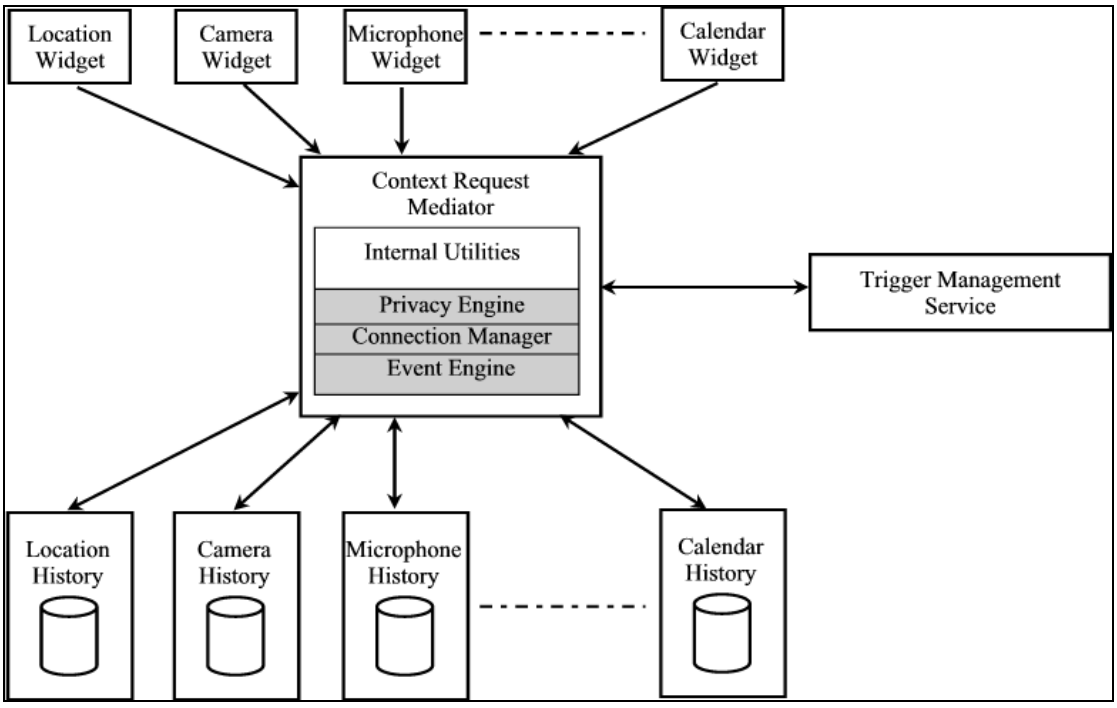

Figure 4.3: Secure Context Service

Each widget is responsible for collecting information from one particular sensor. Examples of widgets are a location widget which is responsible for determining the user's location; a camera widget which uses a cell phone's camera to determine information about the lighting; or a microphone widget which detects the ambient noise level in the user's environment.

Each history store collects time stamped information about each user's previous context information. This can then be used to determine extra details that would not otherwise be available, such as determining whether a user is walking or driving based on the rate of change of the user's location.

The Context Request Mediator is responsible for handling all requests for a user's context. It determines which widgets are applicable to the user and combines all the separate widget information into a single, high level context. It also populates the history stores with each user's context, maintains all widget connections and checks for privacy and security conditions before passing a user's context to the Trigger Management Service.

The Trigger Management Service (Figure 4.4) is the module that initiates the notifications based on events. The Trigger Management Service consists of a Trigger Manager, an Event Matching Engine and a set of Data Stores. The Trigger Manager maintains a list of Triggers that are running. Each Trigger is associated with a particular event or content type such as new email arriving on the mail server, or a web site being updated.

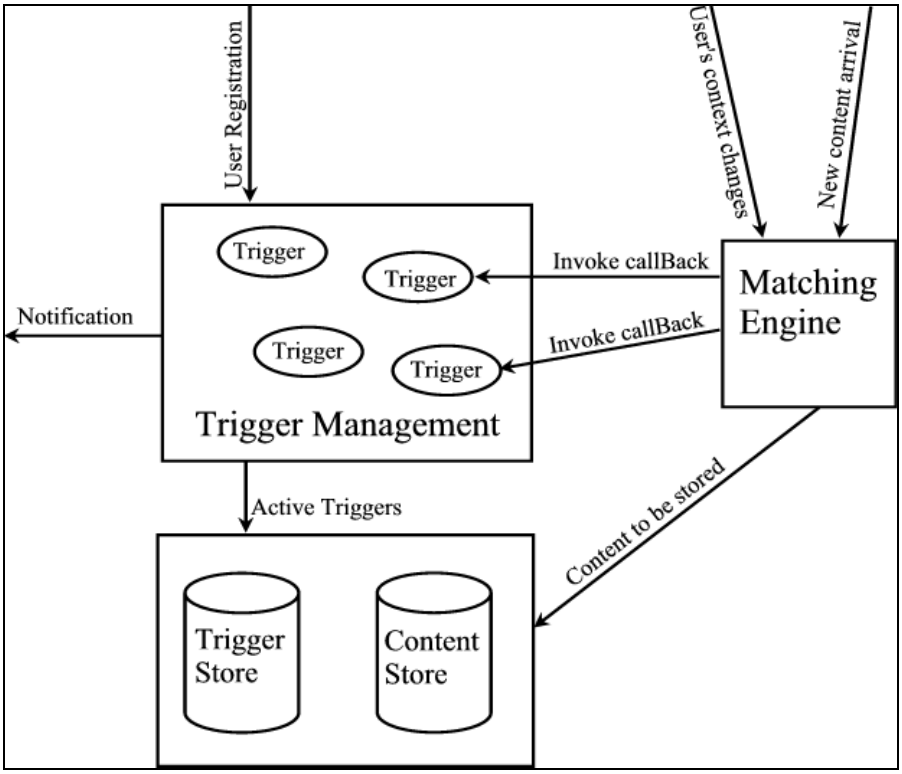

Figure 4.4: Trigger Management Service

The Matching Engine receives an alert about the event that has occurred, as well as the information about the event. It then checks the Trigger Manager to see if there are any Triggers running for the event. If there are, the Matching Engine writes the event and its data to the Content Store and calls the specific Trigger's *handleMatch* method. Each Trigger's *handleMatch* method could perform certain tasks; generally it will result in a notification being made, but it could also remove the Trigger from the Trigger Manager. This is to cater for those events that a user only wants to be notified of once such as stock market prices. If a user wants to be notified if a stock price drops below a certain value, he/she might not want to be notified continuously if the price keeps dropping once it has reached that point. The Matching Engine could also respond to changes in a user's context. If a Trigger is dependent on a user's context, the Matching Engine will be alerted to the user's change in content and check the Trigger Manager for any Triggers that require this information.

Upon user registration, a set of Triggers are generated based on the services and preferences that the user specifies during registration. Each of these Triggers is added to the Trigger Manager and has its status set to Running. Every Trigger that is running in the Trigger Manager is stored in the Trigger Store. This helps to make

recovering the server an easier process as all Triggers that are running can be reloaded easily. If a Trigger's *handleMatch* method results in a notification being made, a request is placed on the Output Module's Notification Request Queue (Figure 4.5).

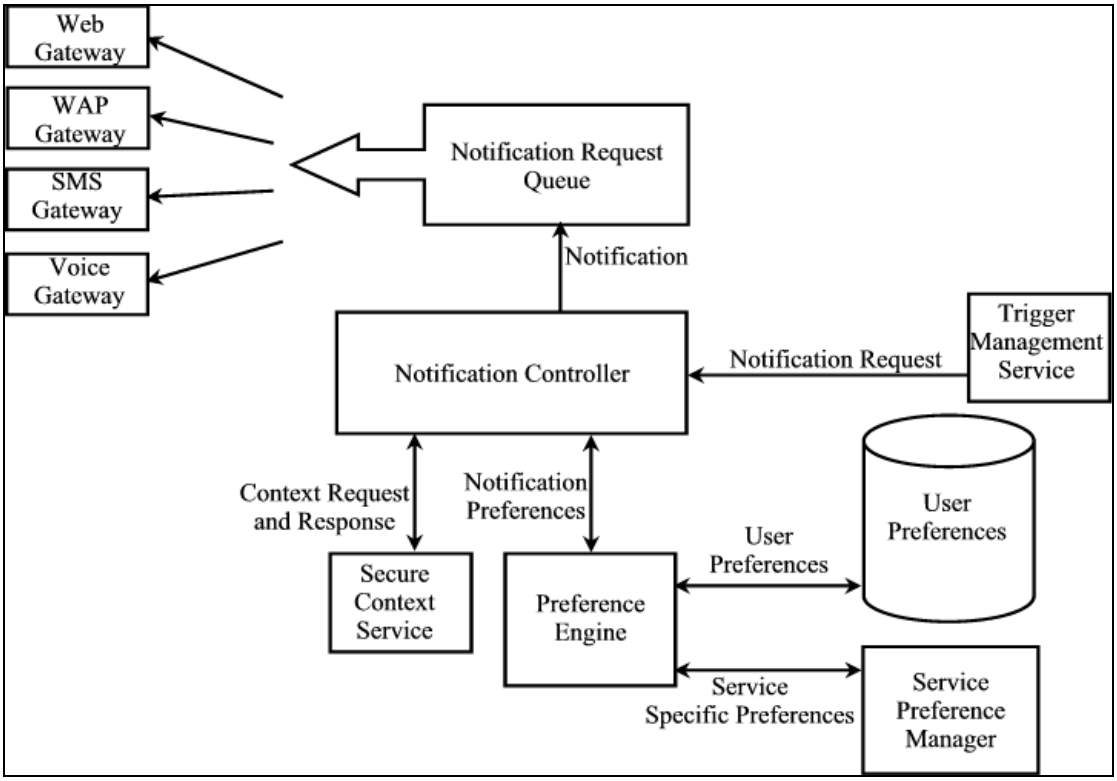

Figure 4.5: Output Module

The Output Module (Figure 4.5) consists of a Notification Controller, a Preference Engine and a Notification Request Queue. The Preference Engine consists of a data store containing each user's generic preferences (for example, if user A is in context B, notify using mode C), and multiple Service Preference Managers which hold notification preferences specific to each service. This allows services like an email notification service to include preferences specific to the sender of the email. This allows notifications to be filtered based on content type and not just on their occurrence.

When the Notification Controller receives a notification request from the Trigger Management Service, it requests the user's context from the Secure Context Service and the user's preferences from the Preference Engine. Once it has this information,

it makes a decision on the modes of output. It then places the notification on the Notification Request Queue along with the output modes.

The Notification Request Queue selects the message in the queue and, based on the modes of output required and the devices available, it selects the appropriate gateway(s) to dispatch the notifications. Each gateway that has been selected then sends the notification to the user.

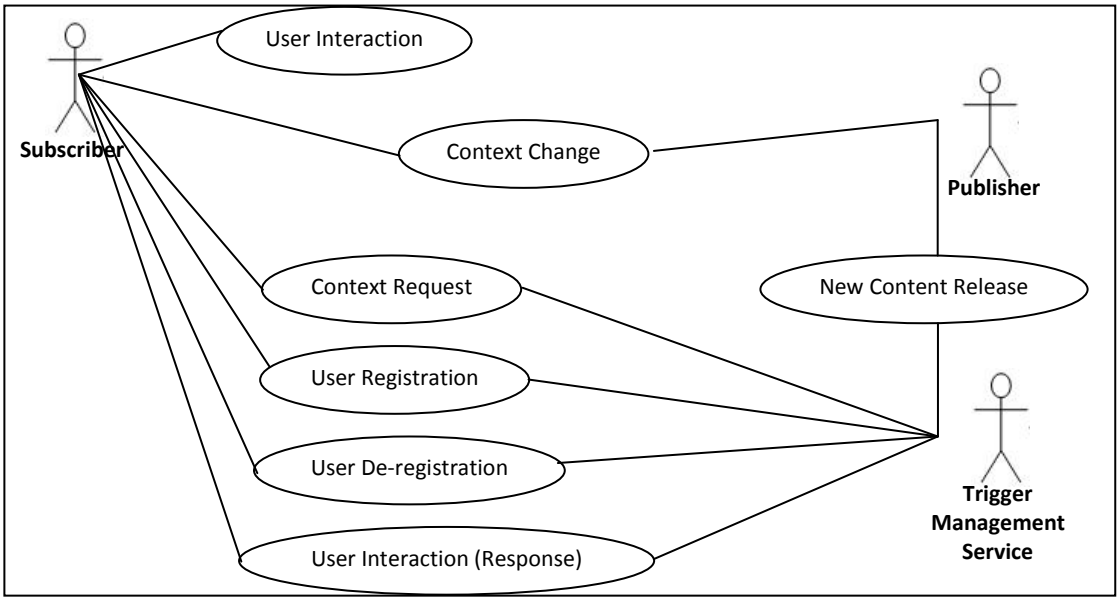

Figure 4.6: Interaction Use Case Diagram

4.3 Functional design

There are seven events that can occur during operation of the proposed model indicated as A to F in Figure 4.1. The main events are new content arrival, user registration, user de-registration, user interaction, a change in a user's context and a request for a user's context. Each of these events results in a different sequence of interaction between the various modules of the model.

4.3.1 New content received

When new content is received (A in Figure 4.1), the Content Retrieval Servlet discovers this by either a push or a pull-based approach. It then retrieves the content and passes it to the associated Content Adapter. The Content Adapter then

encapsulates the content, parses it into a suitable format and then sends it to the Trigger Management Service.

The Trigger Management Service then uses the Matching Engine to see if there are any Triggers associated with the new content. If there are, the content is stored in the Content Store and the *handleMatch* methods for the associated Triggers are invoked. The *handleMatch* method is a method which executes various actions based on the Trigger type. If the *handleMatch* method indicates that a notification must take place, a notification request is passed to the Output Module along with the content of the notification that must be made.

The Output Module receives the notification request and requests the associated user's content from the Secure Content Service. The user's context includes information about the devices available for notification. Once the context has been retrieved, the Output Module retrieves the user's preferences and service preferences from the Preference Engine. Output modes are chosen based on the user's preferences and current context and these modes and the notification to be made are placed on the Notification Request Queue. The Notification Request Queue then selects the appropriate Gateways for the required modes and sends the notification through each of these Gateways. Each Gateway must provide its own methods for converting the raw notification message into the correct format for notification.

4.3.2 User registration / update details and preferences

User registration (B in Figure 4.1) is split into several sections. First the user is asked to complete personal information. The user is then asked to select for which services he/she wants to register. The user is then asked for any parameters required by the selected services. Once all the information has been collected, the Triggers associated with the selected services are added to the Trigger Management Service. The user then supplies his/her notification preferences for each context that he/she could be in when a notification occurs. This information is stored in the Data Store of the Secure Context Service.

4.3.3 User de-registration

When a user wishes to de-register from a set of services (E in Figure 4.1), he/she must select the services he/she wishes to de-register. The associated triggers are then removed from the Trigger Management Service but all parameter information is kept. This is to make it easier for users to re-register for services as all their previous information is still stored.

4.3.4 User interaction (in response to a notification)

When a notification is made (C.2 in Figure 4.1), the user is able to respond to that notification in several ways, depending on the service with which the notification is associated. If the user wishes to respond to the notification, a list of applicable options is presented to the user. For email notification, these options could be to reply to an email, to delete it, to read the entire message, to forward it, etc. Each option is managed by a Response Servlet that is implemented for that service. If there are no Response Servlets that apply to that service, no options are presented.

Once the user has selected the option he/she wishes to perform, the servlet associated with the option requests any additional parameters required for that option (this could be the intended recipient in the case where the user wishes to forward an email). Once the required parameters are completed, the servlet executes and the response is shown to the user. The response could either be a confirmation, a warning or any other information that the user should see.

4.3.5 User interaction (not in response to a notification)

User interaction (C.1 in Figure 4.1) does not need to be in response to a notification. In the case of an email notification service, the user might want to check his/her email. In this case, the user is presented with the list of Request Servlets available for all the services for which he/she has registered. The selected Request Servlet will then ask the user for any parameters required in order to perform the selected operation. Once the parameters have been collected, the servlet will perform a series of operations. There could be many different operations with multiple parameter requests from the user at each step. This is to allow for continued interaction without having to re-select each option. An example would be if the user wishes to check

his/her email. The user's Inbox would be shown and the user could select an email. Once finished with that email, the user would be returned to the Inbox.

4.3.6 Secure context service requires user's context

When the Secure Context Service receives a request for a user's context (F in Figure 4.1), the Secure Context Service consults the Privacy Engine to see what information the user wishes to make available.

The information that is used to determine a user's context can take several different forms. Generally speaking, one sensor is used to detect a particular piece of information. Each piece of information can then be used to determine the user's context. A brief overview of possible sensors and the information each sensor provides, is given in Section 3.3.1 (Table 3.1).

Each sensor is implemented by a specific set of widgets. A widget is a piece of software that runs on the user's phone and provides the information from each sensor. A camera widget would, for example, provide optical information from the phone's camera (if it has one).

Once the Secure Context Service has determined what information the user wants to be made available, the widgets associated with that information are activated. Each time a widget is activated, its result is placed in a Content Store associated with that widget. This is only done if it makes sense to store history information about that widget. For example, it does not make sense to store time history, as the time will be the same as the timestamp associated with it.

Each widget could be running at a different location with a different interface, and therefore a Connection Manager is used to maintain connections with the various widgets. If a widget is unavailable for some reason, it is ignored and the context is determined based on the remaining widgets.

Once all available widgets have returned their values, the Context Request Mediator combines all this information to provide a single context (in a meeting, sleeping, etc). In order to determine a user's context, it is necessary to specify what each possible

context is and what it means to be in that particular context. These possible contexts are stored in the Context Table (Table 4.1) along with the weighting contribution that each sensor makes towards determining that context. If a new sensor is added, the weightings that determine each context need to be adjusted, but this does not mean that a sensor has to be used when determining a user's context. For example, the user's current location is not used when determining whether the user is driving, whilst the rate of change of the user's location is important in determining this.

	Context	Light	Sound	Location	Speed	Calendar	Total
1	**At Desk**	5	5	45	30	15	100
2	**Sleeping**	10	10	15	10	55	100
3	**Home (Other)**	10	10	15	10	55	100
4	**Work (Other)**	5	5	35	35	25	100
5	**Driving**	5	10	0	80	5	100
6	**In Meeting**	10	20	5	15	50	100

Table 4.1: The context table

A user's current context is determined by taking the combined input from the various sensors available to the system and making an inference based on the results. Each sensor has a different level of importance (or *weighting)* for each particular context. Driving, for example, uses more information about the speed of the user than the light surrounding the user. The weightings are then used to determine a users' context using cues as described by Schmidt *et al* (Schmidt *et al*, 1999b)

As soon as a user's context is requested, the Context Module connects to the database and reads all the possible contexts and their associated weightings. The system loops through each of the possible contexts and then checks the confidence level (C_i) of the user being in that context. The context with the highest confidence is taken as the user's context. The confidence is calculated by taking the sum of all the weightings of the sensors (W_i) multiplied by the value (V_i) of that sensor (Figure 4.7). If a

particular device or sensor is not available, the weighting of each of the remaining sensors is increased by the ratio of that sensor's weighting to the remaining sensors.

$$C_i = \sum W_i V_i$$

Figure 4.7: Formula to determine the confidence of a particular context

For example, if a message is sent to user Ryan, the Context Module requests Ryan's widget information. Once all the information is returned the Context Module sums the values combined with the weightings to reach a total confidence (C_i) that Ryan is in each context (Table 4.2). Once all the contexts have been tested, the context with the highest confidence value is taken as Ryan's context, namely At Desk.

Context Description	Confidence Value
At Desk	64
Sleeping	8
Home (Other)	16
Work (Other)	54
Driving	3
In Meeting	49

Table 4.2: Context test example

The resulting context is then passed back to the Trigger Management Service. The impact of context on output mode was discussed in Section 3.3. The suggested contexts were determined from literature (Abowd *et al.* 1999). These contexts are discussed further and evaluated in Chapter 6.

4.3.7 User's context changes

If a change is detected in the result of a widget (the user's context has changed), the change is stored in the Content Store associated with that widget. If the user is currently interacting with the system and the change in context implies a change in mode, the mode is not immediately changed. Once an action is initiated, the output mode stays the same for the duration of the action. It may not be possible to constantly monitor widgets in order to detect changes as accessing certain widgets

might incur a cost to either the system or the user. As a result, each user should be able to specify whether or not continuous tracking should be used for each widget.

4.4 Data Design

Data design shows what information must be stored and how the data relates to other data. The data must be designed in such a way that it is flexible, non-redundant and allows for all the required tasks to be performed. The data design of the proposed model has been split into two sections, namely the notification data design and the service data design. The notification data design shows the data involved in a notification/registration independent of which service is being implemented and the service data design shows what data is required in order for services to be implemented.

The Class Structure Diagram for the notification data is shown in Figure 4.8. The User class stores all personal user information such as the user's name, surname, contact details etc. Information related to services is stored in the Service Parameters class. This could be information regarding the mail server, user codes, passwords, POP3 details, etc. The Service Parameters class is sub classed into one class per service. This is to allow for different numbers and formats of parameters for each service.

The Service class contains information on all services provided. It includes a unique identifier, a name and a short description of the service. The User Registration class contains information on users that have registered for specific services. It is an association class and only contains the service ID and the user ID. By forcing each service to implement its own Service and Service Parameters class additional services can be added to the model without any changes being required. Services can also be removed and changed without affecting the remaining services.

The Content class stores the content that is received from the content adapters. This information can then be accessed by users or notifications at a later stage. The class is sub classed into as many subclasses as there are different services. This allows for different types of content with different formats to be used for notification purposes.

The Trigger class stores the trigger which fires on an event. Each instance of this class has a *handleMatch* method which gets fired when there is a match between a trigger and an event. Each trigger is also associated with a particular service and user. The relation to user is shown in the User Trigger class which is an association class representing the many-to-many relationship between the User class and the Trigger class. Each service can have many Triggers classes associated with it. This means that triggers can be added or removed to services without affecting other services.

The Widget class is the class that runs on the mobile phones collecting information about the user's context from the various sensors. Each widget is responsible for getting information from one sensor on a device. This means that a phone may have multiple widgets. This also means that new devices and sensors can be supported by implementing a new Widget for each new sensor or a Widget for each sensor on the new device. The Widget class is sub classed into various other widget classes, one for each sensor type. Each sub class overrides the *getValues* method so that different methods can be used for different sensors. The User Widget class is an association class that represents the many-to-many relationship between the User and Widget classes.

The Context class contains information on possible contexts. A list of applicable contexts was taken from literature and determined by the environment and purpose of the prototype (Abowd *et al.* 1999). These contexts are the contexts which would fit most of the environments that the users would typically be in (Table 4.2).

Each of these contexts implies that notification could take place in a different output mode, depending on each user's preferences.

The User Context class consists of a number of User Context values. This is a list of all widgets that the user's mobile device supports. Based on the values of these widgets, a context is chosen from the Context class for the user.

The Mode class is a list of possible output modes. The list of output modes is based on the capabilities of the target devices. Cell phones were selected as the target

device for this research (Sections 2.2 – 2.4). This limits the available output modes to the modes supported by cell phones. This can be changed at a later stage, however by implementing the appropriate Mode classes. For this research, the output modes are limited to the following:

1. Text;
2. voice; and
3. text and voice

The Preferences class stores all the user's preferences. It contains the ID of the user, the ID of the context and the ID of the mode that the user would prefer to use for the given context. The Default Modes class contains a list of values for each context that is placed in the Preferences class during registration. The user can either accept the default values or edit them.

The Gateway class is the parent class of all output gateways used for notification. Each gateway is a subclass of the base Gateway class. This enables new gateways to be added by sub classing the Gateway class and registering the new gateway with the system. Each gateway has a *sendMessage* method which sends a message to a user. Each sub class of gateway has its own implementation of this method. The Gateway Mode class contains information showing which modes each gateway supports.

The Notification class contains information on all notifications made, the gateway used for the notification and the trigger which caused the notification. The content of the notification is also stored.

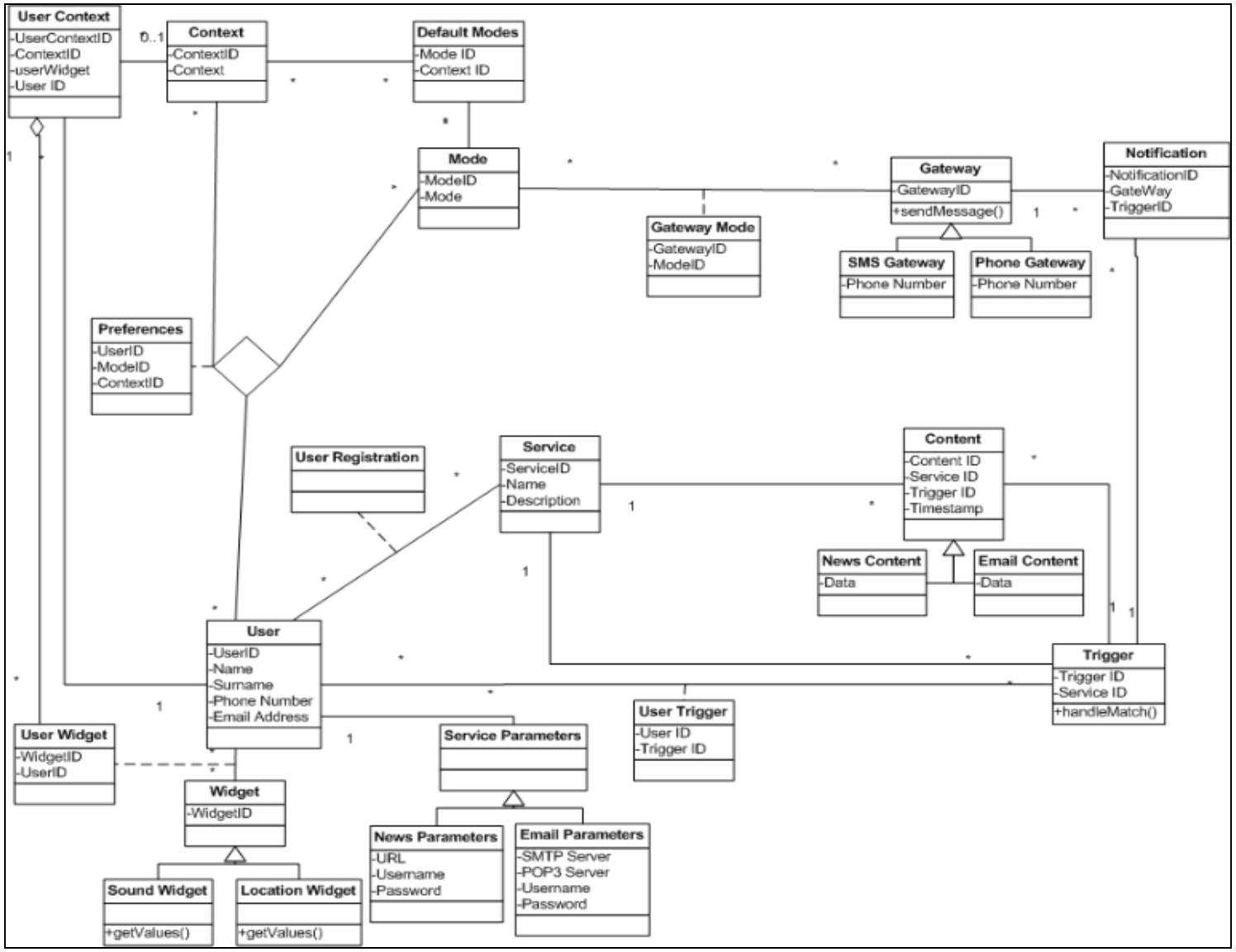

Figure 4.8: Class structure diagram for adaptive multimodal mobile notification service

The Service Data Design (Figure 4.9) shows how the classes of each service relate to each other in order to create and add a new service for notification. Each service is made up of a number of difference servlets. Each servlet can either be a Setup Servlet or a Details Servlet. The setup servlets are used to get the parameters and any other details needed in order to configure the service for each user. In the case of an email service, the Setup Servlet would get the user's username and password for the mail server as well as information on how to access the mail server. Once the Setup Servlet has retrieved all the parameters, the values retrieved are stored in the associated Service Parameters class.

The Details Servlets are further sub classed into either a Request Servlet or a Response Servlet. Response Servlets are classes that can be executed in response to a notification. If a user is notified of an email, they can choose to delete or forward the email. Any parameters needed to complete the operation are then requested (such as email address of person to whom email should be forwarded to) and the operation is performed. Each action will be implemented by a single Response Servlet.

Request Servlets function similar to Response Servlets but do not occur in response to a notification. These servlets could be used to compose a new email, check the Inbox, etc. When the user selects a Request Servlet any parameters needed (recipient, message body) are requested and the action is then performed.

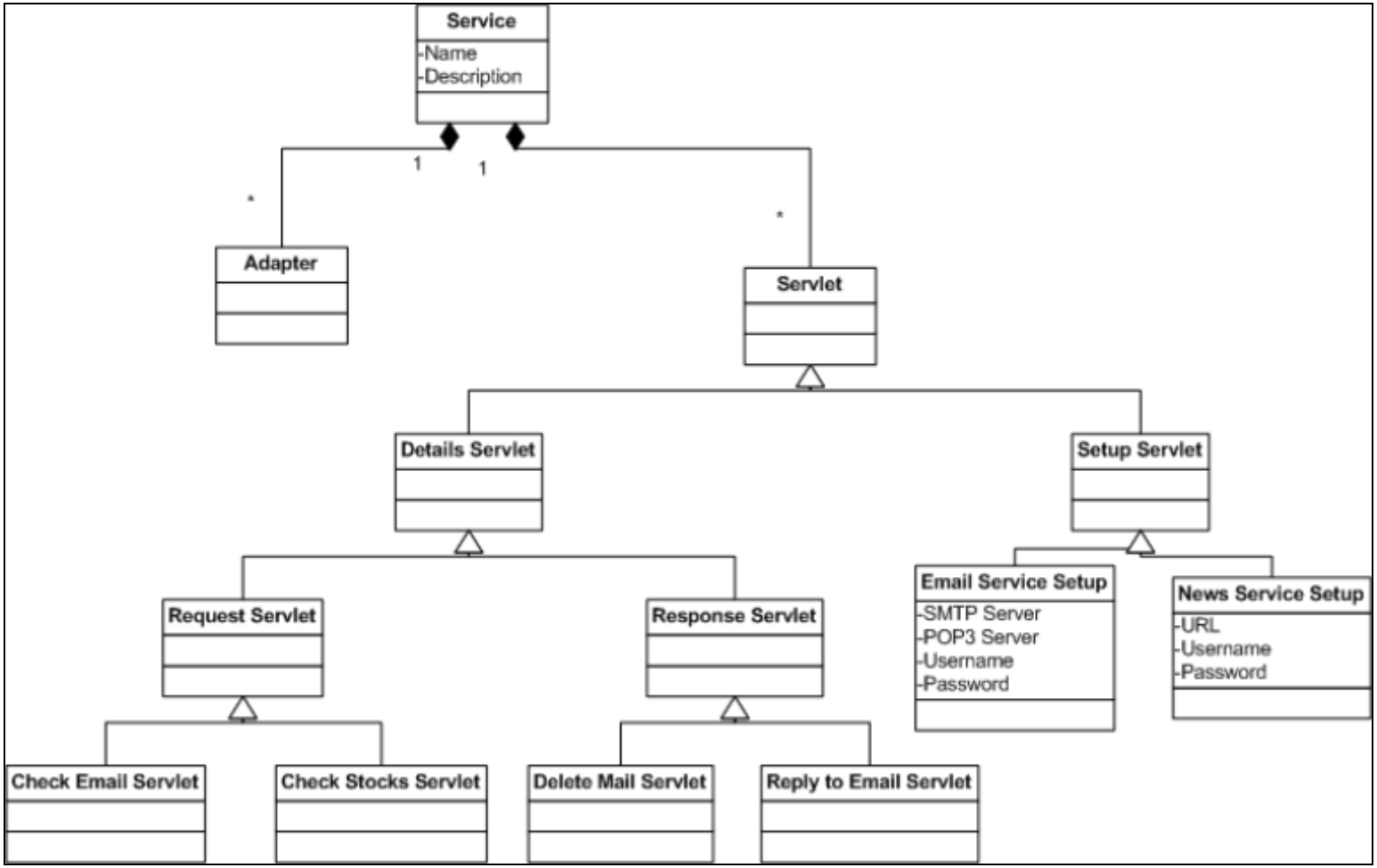

Figure 4.9: Service data design

4.5 Interface Design

One of the first steps involved in designing the user interaction is to understand what requirements each interface must satisfy. There are four main tasks that a user performs during interaction with an application developed using the proposed model, namely Registration (Section 4.3.2), De-registration (Section 4.3.3), Reading a notification (Section 4.3.4) or requesting/responding to a notification (Section 4.3.5 and Section 4.3.4).

Figure 4.10: User registration form

During registration (Figure 4.10), there are various items of information that is required from the user, namely the user's personal information, any parameters needed by the service for which he/she is registering as well as the user's context preferences. Personal information contains data about the person, including first name, surname, contact information, etc. Service parameters comprise information needed for the service such as the mail server network address, user code and password, etc. Context preference information allows the user to choose what mode

he/she would prefer in certain situations. For instance, if the user is in a public place would he/she prefer text or voice? The process of changing a user's preferences or details is almost identical to registration except there would be previous information already filled in instead of having to enter everything from scratch.

Figure 4.11: De-registration form

De-registration (Figure 4.11) is a simple process whereby the user selects the services which he/she wishes to de-register. Thus the information needed on this screen is a list of services which the user is currently registered for and a deregistration confirmation.

When new information arrives, the objective is for the user to either receive notification about new information, to receive the information itself, or to not receive a notification at all. An example of notification of new information would be the case of email notification. In this case, the header of the email would be sent to the user and if the user wanted to view the contents of the email, he/she could request it. Alternatively, notification of stock market prices could notify the user of a change in stock prices as well as new prices.

User interaction can either be in response to a notification, or a request for a notification. In both cases the options available for the user are service specific. For an email notification, options could be deleting email, forwarding the email, composing a new email, etc. Each servlet that performs a task will be responsible for providing the fields required for the user to complete the task.

4.6 Conclusions

The IBM Intelligent Notification Service model was adapted to include more user interaction and context awareness (Section 4.2). This resulted in the inclusion of two additional modules, namely the Input Module and the Output Module. The flow of data between the existing modules was also changed (Figure 4.1).

Seven key events were identified, namely new content received, user registration, user de-registration, user interaction in response to a notification, user notification that requests a notification, the Secure Context Service requesting the user's context and when the user's context changes (Section 4.3).

A formula was determined to assign appropriate weightings to determine a user's context based on sensor information. This formula is used to determine the user's context which is then used with the user's preferences to determine the output mode (Section 4.3).

A class structure diagram was created to show the various classes required to support adaptive multimodal mobile notifications (Section 4.4). The class structure diagram shows that the model is completely extensible allowing for plug-ins and changes at almost all levels.

An outline of the information needed for each task was presented and initial user interfaces were designed to allow users to interact with an application developed using the proposed model on small screen devices (Section 4.5). To encourage a similar look and feel of all user interfaces, each servlet merely requests certain information and the application itself is responsible for how the request is shown.

The development of a prototype to demonstrate the effectiveness of the proposed model is discussed in the next chapter.

Chapter 5 Prototype Implementation

5.1 Introduction

This chapter aims to answer research question 5 (Table 1.1). A full prototype of the model was implemented as a proof-of-concept. The prototype was given the name Mercury after the winged messenger of the gods from Greek Mythology. This chapter documents the implementation process. Mercury was implemented in four distinct modules, namely an Input Module, an Output Module, a Trigger Management Module and a Context Module. Each module was designed to perform a specific task but to be independent of other modules except at the start or end of its process flow. This design made it easy to split the implementation into manageable sections.

The rest of this chapter discusses the details of the implementation of Mercury. Each module will be discussed separately starting with the implementation of the Data Store, the Input Module, the Trigger Management Module, the Output Module and finally the Context Module. Following the discussion of the different modules a brief explanation of how the modules interact with each other is given.

5.2 Services Implemented

The scope of the research was limited to notification of emails (Section 1.4). Mercury, however, was implemented to notify users of emails as well as network state changes in order to prove that additional services could be included in applications developed using the proposed model without any redesign. The initial implementation included only email notification and network state change notification was added after the initial deployment.

5.3 Data Implementation

The database was implemented in Microsoft SQL Server 2005 following the data design specified in Section 4.4. There are a few concerns when implementing databases for use in mobile environments. Firstly, mobile devices have limited

bandwidth and connectivity (Afonso *et al.* 2004) and as a result, data communication between the device and the server needs to be limited to absolute essentials. Another requirement is that the server needs to respond quickly to maximise the chance of the data being transmitted without the connection dropping.

The model was designed to notify individual users in the most suitable mode, and as such, each user's personal details, preferences and settings need to be stored by Mercury. Personal details include the user's name, surname, cell phone number, fax number and email address. Personal details can be statically defined as they are the same for each service; preferences and settings are unique to the type of service for which the user is registered, and the type of devices the user has available for notification.

User preferences are stored as a link between the mode of notification and the user's context. This means that if a user is in a particular context, he/she wants to be notified in a particular mode. A single-layered neural network was implemented in order to keep the user preferences up to date. The neural network is based on a typical pattern recognition neural network (Figure 5.1). After registration, the neural network uses the set of initially chosen preferences as the base training case and training continues for a period of two weeks. During this training period, every time a notification is made, the user is asked if the notification was made in the most appropriate mode. If not, the user is asked what the most appropriate mode should be. The answers to these questions are fed back into the neural network which then readjusts the weightings for each mode. This has the affect of changing the user's mode preferences with actual field usage scenarios. This training approach helps address the fact that most users are incorrect in their judgement of what the correct mode should be (Chen and Kotz, 2001), by learning from example to determine what the user really wants.

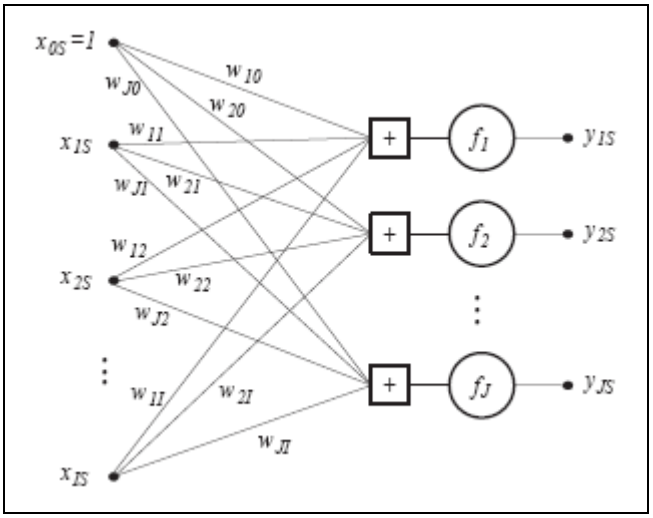

Figure 5.1: A single layer neural network for pattern recognition (Chen and Kotz, 2001)

Service settings are unique to a user and a service. In order to allow for extensibility, the service settings are not hard coded into the prototype. Each service is responsible for managing its own storage of user settings, maintaining a list of settings that are required from the user and being able to identify a particular user's settings from a User ID. In order to facilitate this abstraction, each service requires its own Setup Servlet which is responsible for returning an array of all settings that are required for the service as well as saving those settings in a data store. The Setup Servlet is discussed in more detail in Section 5.4.

Each user's contextual history is stored to facilitate time-based inferences from the user's context. This historical information can be used to determine additional information from existing contextual information. An example would be to determine a user's speed from the history of the user's location.

In order to improve efficiency, a small amount of data must be transmitted to and from the mobile device and database connections should be as few as possible. This becomes a problem when there are a variable number of parameters needing to be sent to the database as SQL does not allow for arrays or variable amounts of parameters. To overcome this, all parameters are concatenated in a string and a user defined SQL function is used within the stored procedure to split the string back into its original parameters. This means that only one connection is needed to the database instead of one per parameter.

5.4 Input Module

The Input Module is responsible for taking input from multiple devices and modes and interpreting the input into a command ready for use within the prototype. For the purpose of this research, a small selection of devices and modes were chosen as the input types to support. The devices chosen were laptops, WAP enabled cell phones, PDAs and desktop computers. Each of these devices provides similar information, but does so in different ways using different formats. For example, a cell phone can phone in and use voice to provide input while a desktop computer can log on via the internet and use Voice Over IP for voice input. This makes it necessary to implement a separate gateway, recognition and interpreter component for each device and mode.

Only one device was catered for during the initial implementation phase. This was done purposefully so that the extensibility of the model could be tested by adding support for the remaining devices after Mercury was completed. The initial device that was supported was a cell phone, and the input modes supported were text and voice. The user could use his/her cell phone to provide textual input via the phone's WAP browser and vocal input by phoning Mercury and speaking commands to it. Multimodal input over the cell phone is limited to one mode per command; therefore if the user starts a command using voice, the user cannot switch to text until that command is completed. This limitation is only for cell phones and PDAs while the rest of the devices can switch modes as many times as required during input.

For textual input from a cell phone and PDA, .Net Mobile was used to create the interface forms. .Net Mobile generates the tags required to render the form based on the device that requested the form, which means that each screen could be rendered differently based on the mobile device although the same information and inputs are being rendered. ASP .Net was used for the textual input for laptops and desktop PCs. Due to the almost identical way in which these two devices access the Internet, there was no need to make a separate gateway for each of these two devices (Section 4.2).

Voice input was supported by using the Microsoft Speech SDK. Users could use a cell phone to phone a modem and speak instructions to Mercury. These instructions

would then be interpreted by the Microsoft Speech SDK and translated into commands. These commands are passed to the Trigger Management Module for further processing (Section 4.2). Laptops and desktop PCs can connect to the Internet and use Voice Over IP and a microphone to speak instructions to the server. This implies the need for a fast, stable connection to the Internet. The server interprets the instructions using the Microsoft Speech SDK, translates them into commands and passes this information to the Trigger Management Module. Voice input from a PDA is not supported unless the PDA supports GSM network access, in which case, the PDA is treated the same as a cell phone.

User input is required in order to setup each service correctly; this was made extensible by requiring that each service provides its own Setup Servlet which has the responsibility of providing information about the parameters required by the service.

The Input Module uses late binding to load each service's Setup Servlet class from a strictly named dynamic link library (DLL) and class (class name *ServiceNameSetupServlet*). Each DLL provides public access to the *getParameters* and *setParameters* methods. Late binding was used so that a DLL could simply be placed in the system's *Gateways* folder and the prototype would be able to use it without having to be recompiled or restarted. Late binding is performed by loading a DLL at runtime, as opposed to referencing one during compile time, and calling a public method exposed by the DLL.

5.5 Trigger Management Module

The Trigger Management Module (Section 4.2) accepts inputs from the Input Module, the Context Module, as well as from the various content sources. Input from the Input Module can either be used to update a user's profile, request a notification or respond to a notification. Input from the Context Module is not used to determine the mode output (that function lies within the Output Module), but is used to trigger any notifications that are based on a user's change in context.

User profile updates can either require updating internal or external user settings. Internal user's settings are settings that are independent of any content sources, such

as contact details or device settings. External details are those details that are required for a specific service such as an email account password or stock codes to monitor. Updating internal details is a simple matter of updating the specific fields in the Trigger Management Module's database to reflect the changes. External details are updated using each service's Setup Servlet. Once the service has provided a list of inputs and their types, the Input Module renders these components based on the device and mode required. When the user completes input, the details are passed to the *saveDetails* method which saves the details. Each service can save the data in whichever format it wants, namely in flat files, XML or in a database.

When a notification has been made, the user who receives the notification will have a chance to respond to the notification. An example would be if a user receives an email, the user should be able to reply to the email or forward it etc. Each service will have its own various response types, possibly unique to that service and notification type. A notification regarding a change in price for a particular stock has no meaning for a "Reply To" option which an email notification might require. This makes it necessary for the options available for a user to respond to a notification, to reside within the Service itself rather than the Trigger Management Module. In this case, user input follows the same process as for external detail where the service provides a list of input requirements and types to the Input Module for rendering. A hidden type is also passed with the data to the Input Module in order to link the input from a user to a prior notification. This link to existing notifications makes it possible to perform actions on a notification.

The Trigger Management Module can also receive a request for notification from the Input Module. This happens in the case where a user wishes to check his or her Inbox for emails. Most of these types of interaction will require a response to the notification, i.e. once the user has a list of emails in his or her Inbox, the user might want to delete or reply to certain messages. This request and respond interaction is supported in the same way as if the user were responding to a previous notification, except that the hidden type is used to link a notification to a particular item within the Service and not a previous notification.

The Trigger Management Module's main function is to respond to updates within the many Content Sources. This allows for notifications to be made when new information is available for the user. Each Content Source could have a different method for checking for updates. Some of these Content Sources could be checked by viewing a RSS feed (an example would be a stock listing site which provides updates via a RSS feed) to see if there is any information available whilst other Content Sources could require requesting all the information and then checking if anything has changed (this could be used to check whether any website content has been updated).

Due to the number of possible ways in which content can be checked for updates, it is not possible to create all of the checking functionality within the Trigger Management Module and this function is delegated to each service. This implies that, if a service is responsible for notifying a user of new emails, the service is also responsible for checking the users' Inbox. Placing this function as a Servlet within the Service allows for complete extensibility with regard to content sources. The Trigger Management Module does not require any information about how the information was obtained, only that there was new information received and the user must now be notified.

A change in a user's context could trigger some form of notification. As a result, a standard Servlet is created within the Trigger Management Module which has the role of monitoring a users' context. Once a context change has occurred that meets certain requirements (the user changes from a driving context to a having lunch context) an action is performed on the messages currently in the notification queue. This allows for either the notifications being cancelled, or for notifications that are currently pending being sent – depending on what the change in context means for that user.

Performing the actual notification to the user is not a function for which the Trigger Management Module is responsible; this resides within the Output Module. Once the Trigger Management Module receives new content for a user, it places a request for a notification in the Notification Controller of the Output Module. The Output Module then notifies the user in the most appropriate mode.

5.6 Output Module

The Output Module is responsible for sending notifications to the users as well as using the user's context to determine the correct mode in which to send the notification. The Output Module is not responsible, however, for determining the users' context. The Output Module consists of a Notification Controller which receives a request for a notification and places the notification as well as the device and mode of the notification on the Notification Request Queue (Section 4.2).

The Notification Controller is implemented as a service which listens for notification requests on a network port. This functionality allows for a distributed implementation of the model, meaning that the Output Module does not need to be implemented on the same machine as the Trigger Management Module (this was required in the implementation environment due to firewall and security restrictions). Due to the fact that listening for network connections on a port blocks the system, a multithreaded environment was used. Once a connection is received, the Notification Controller expects to receive a 32 bit integer. This integer is the ID of the service causing the notification. Another 32 bit integer is then expected, which is the ID of the user to which the notification must be sent. Once the user ID has been received, the Controller expects a string which is the text version of the message to be sent. Note that the message being stored in a string format has nothing to do with the mode in which the notification will be made.

Once a request for a notification has been received, the Output Module determines to which device the notification must be sent, and based on the device and context, what mode in which to send the notification. The device and mode decision is based on the user's context and preferences. A user preference means that if a user is in a certain context, he/she wants to be notified in a certain mode. User preferences also relate to a specific service. Each user can have his/her own set of preferences for each service; for example to only notify the user of emails from a particular set of contacts, or always use text notification for emails with a subject which contains the string "Free Offer". Each service is responsible for handling the user preferences, including storing the preferences and decision making based on these preferences. Service preferences override user context preferences with respect to the mode of output.

Once the output mode and device has been determined, the Output Module connects to the database and retrieves the address of the user. The gateway required for that particular output is passed the address to which the notification must be sent as well as the message to send. Note that the address for the notification is not necessarily a phone number but refers to a unique destination for the message which could be a phone number, but it could also be an email or an IP address.

Each gateway is responsible for sending a message to a particular device in a particular mode. For example, there would be two gateways for cell phones, one for sending a text message to the phone and another for sending a voice message. Initially these were the only two gateways implemented. This was done to evaluate the extensibility of the Output Module. In order to make sure that each new gateway will not require any changes to the Output Module, each gateway has to implement the *iOutputGateway* interface (Figure 5.5). The *iOutputGateway* interface forces each gateway to have a *sendMessage* method which accepts a string for the destination address and an object for the message to be sent. This message could be a sound file, a text message or a video file.

```
interface iOutputGateway
{
    bool SupportsMode(string Mode);
    bool SendMessage(string Address, object Message);
}
```

Figure 5.2: The iOutputGateway interface

Since each gateway sends messages to a different device, each gateway must handle the transmission of the message. The SMS Gateway uses a Web Service in order to send its messages (Figure 5.6) whilst the Phone Gateway uses a modem connected to the computer running the gateway to obtain a dial-tone and phone the intended recipient.

Each message is sent with a unique number to ensure that users can respond to particular messages in a way that is appropriate to the message type. For example, an

email has options to delete, reply or forward a message. As soon as a user responds to a message, the user input is translated by the Input Module and processed by the relevant service.

5.7 Context Module

The Context Module is responsible for determining each user's context. Each time a notification needs to be sent to a user, the Output Module requests the user's context from the Context Module. The Context Module then determines the user's current context according to the algorithm outlined in Section 4.3.6 Section 4.3 and passes that information back to the Output Module.

5.8 Conclusions

This chapter discussed the development of Mercury, a prototype based on the model proposed in Chapter 4. The proposed model was designed to extend the IBM INS model for mobile notification. Mercury was implemented according to the discussion in Chapter 4. The model did not have to be altered in order to implement the prototype. The implementation answered research question 5 by showing how a prototype of the proposed model could be implemented.

Mercury was implemented to provide multimodal mobile notification to cell phones and laptops in either text or voice (Section 5.4 and Section 5.6). The services implemented were to notify users of new email and of changes in the state of the network (Section 5.2). A pattern recognition neural network was used to allow Mercury to adapt to changing user preferences (Section 5.3).

Laptops, WAP-enabled cell phones, PDAs and desktop PCs were chosen as the devices to support for input and output. The modes supported on these devices were Text and Voice. The implementation also demonstrated the concept of extensibility by adding additional gateways, devices, contexts and services to Mercury after the initial implementation was complete (Section 5.4 - Section 5.6).

The next chapter deals with the evaluation of Mercury and the usefulness of the adaptive modes of notification in order to evaluate the proposed model.

Chapter 6 Evaluation

6.1 Introduction

A model for adaptive mobile notification services was developed and the design outlined in Chapter 4. A prototype, called Mercury, was implemented with the goal of evaluating the feasibility of the model and the implementation outlined in Chapter 5. The aim of this chapter is to determine whether the model satisfies the requirements for adaptive mobile notification. This chapter describes the evaluation of Mercury. The implementation of Mercury was used to gain information required in order to perform a user evaluation. Feedback from questionnaires completed by users of Mercury to obtain information about the suitability of the notifications they received and the choice of mode used by the prototype is discussed in this chapter.

6.2 Model Evaluation

The model proposed in Chapter 4 aims to produce a mobile notification system which adapts to the user's environment by using context awareness and multimodality to switch the mode of notification to the most appropriate mode for the current user based on the user's current context. Chapter 3 reviewed and compared three models for adaptive mobile notification services. In this section, the proposed model will be evaluated against the same set of criteria and compared to the each of the models evaluated in Section 3.6.

There are certain criteria that should be satisfied so that multimodal context aware interaction can take place in a mobile environment (Section 3.4). These criteria are:

- The model must take the mobility of the users into account;
- it must allow for multiple content and interface formats;
- the model must be scalable;
- users information must be kept secure and private;
- context acquisition must be abstracted from context use;
- must be able to include new sensor technology as it becomes available;

- contextual history must be stored;
- must support distributed communication;
- time synchronisation between multiple sensors must be supported;
- must allow for the switching of modes mid-communication; and
- each mode must provide complete functionality without having to rely on another mode.

The model evaluation in Chapter 3 resulted in the IBM INS model being chosen as the model to be adapted. The adaptation was intended to create a model which satisfied all of the requirements for an adaptive mobile notification service. This adaptation resulted in the model presented in Chapter 4. The IBM INS model satisfied all of the requirements except for contextual history use, time synchronization between sensors and functionality of the system independent of mode.

In order to address the lack of contextual history, the proposed model uses the Context Module to determine a user's context. This module relies on various sensors to determine the user's current context. Each of these sensors stores history about the results it has gathered which, in turn, allows the contextual history to be used as a sensor itself. This history sensor allows for contextual classifications that would not otherwise be possible. An example of this would be using the history of a person's location to determine his/her speed. Another use of the contextual history would be for prediction. If a user follows the same contextual pattern each day, the system can use that knowledge as an additional sensor with its own weighting in order to determine the user's context.

Due to the fact that there are numerous possible sensors that can be used to determine a single user's context, there is a need to synchronize the time between the results of each of these sensors. The proposed model achieves this synchronization by using the context request time as the time of the context. This implies that if the module requests a user's location and the process used to determine the location takes five minutes to run, the time associated with that location is the time that the location was requested. This means that when the module requests information from the various sensors, the time associated with all of those requests is based on the module's

internal time and not the time of the sensor and/or the time taken to receive a result. This supports contextual history as there is no guarantee that a sensor will return a result within a practical timeframe (it could take two hours to return a result and the Context Module may need to provide a context within 30 seconds).

The IBM INS model provides user input options in one mode only, namely a web interface. In addition, the model does not allow for inputs from multiple sources and modes to be used for a single command. User input must be completely free of any mode or device requirements and users should be able to switch modes mid-command in order to meet the mode independence requirement of an adaptive mobile notification service. This shortcoming in the IBM INS model has been addressed in the proposed model by means of including an Input Module which has the responsibility of managing all possible user input. The Input Module takes input from various sources and modes, represents this input using a formal grammar, and parses the input into a command. This allows commands to be entered in multiple modes and from multiple devices.

Modifying the IBM INS model to include contextual history, time synchronization and mode independence enables the proposed model to satisfy all of the requirements for an adaptive mobile notification service. A prototype implementation of the model was used to gather user feedback and determine the effectiveness of the model. The following section details the user evaluation and results.

6.3 Usability Evaluation

An adaptive mobile notification service focuses on notifying users of certain events. This implies that such a service requires user interaction and must provide some form of user satisfaction. Usability evaluation was used to gauge this satisfaction and determine the overall acceptability of the prototype.

6.3.1 Goal of evaluation

The goal of the evaluation was to investigate the effect that Mercury has on the working practices of the users and to determine whether adaptation improved the effectiveness of mobile notification in a real-world environment.

6.3.2 Evaluation Questions

In order to achieve the goal of the evaluation, several questions were identified:

- Was the correct context determined?
- Was the correct mode of notification used?
- Were the notifications beneficial?
- Did the users feel comfortable with the level of control they have when using Mercury?
- Did the users have any concerns about being interrupted when using Mercury?
- Did the users have any privacy concerns when using Mercury?

6.3.3 Evaluation Methods

Usability evaluation for desktop environments is not well adapted to mobile environments due to the lack of situational influences (Po, 2003). Steven Garzonis (Garzonis, 2005) showed that field testing is more effective for identifying issues related to the context of use whilst lab testing highlights interface problems which do not hinder the use or effectiveness of the system. It was decided that an extensive field test would be performed spanning a period of two months with several evaluation questionnaires to be completed per user during the period. Each user completed a questionnaire as well as a brief interview every three weeks. The questionnaire can be seen in Appendix A.

6.3.4 Selection of Participants

In order to perform a credible usability evaluation of the prototype, a sample population which accurately reflects the user demographics was selected and used in the evaluation. The prototype was used in a corporate environment with mostly

young executives making use of it to provide them with notifications of new emails, network state changes and notifications of their teams' current function. There were a total of 74 users, all of whom were between the ages of 23 and 31 (Figure 6.2) with 58 percent of them being female (Figure 6.1).

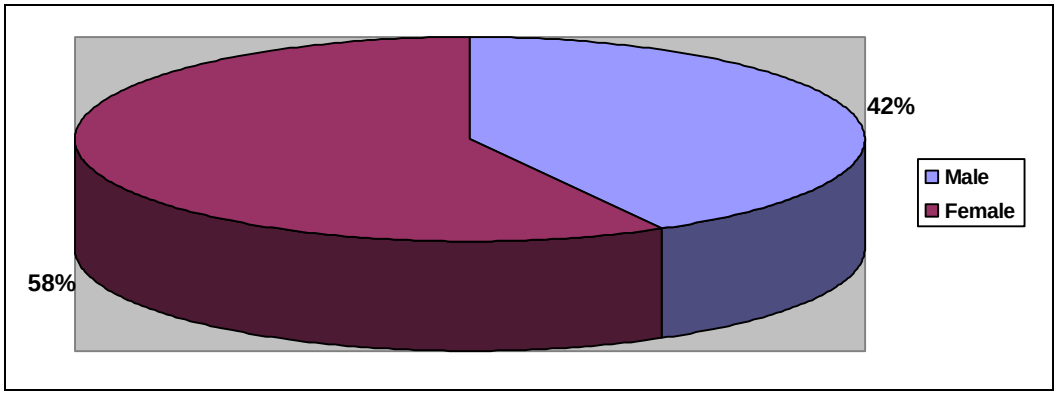

Figure 6.1: Gender profile of participants (N = 74)

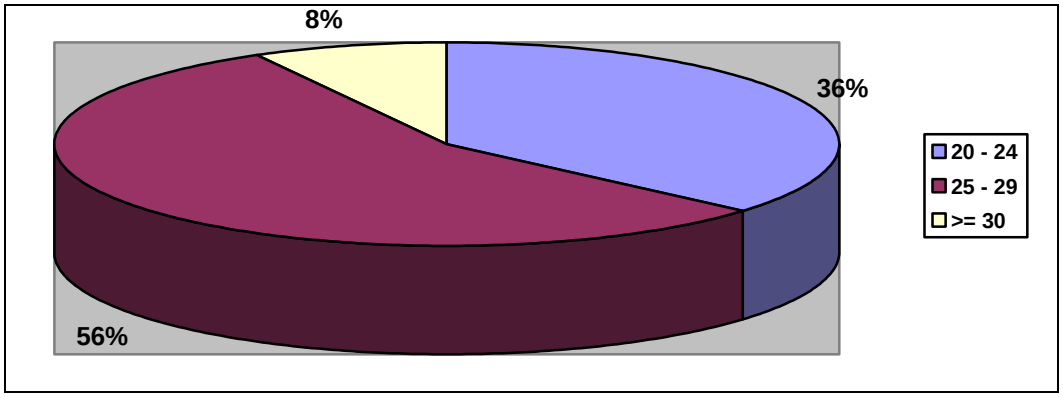

Figure 6.2: Age profile of participants (N = 74)

6.3.5 Tasks to be Performed

The evaluation process was intended to discover how easily Mercury could fit into the users' lives and how accurate Mercury was at choosing the right mode of communication based on the users' context. Upon registration, each user was asked in which mode of communication they would like their notifications to take place. This was stored in the preference table and was used as the initial training for the context module's neural network (Section 5.3). For a period of two weeks after the initial registration, the users were asked whether or not the notification was made in the correct mode. Each response was used to further train the neural network. Once

the training month was completed, the following month was used to determine the usefulness of the trained adaptive mobile notification service.

6.3.6 Results

Users were requested to select their preferred mode for each context during registration for a service. These initial preferences revealed that 84% of users wanted all of their notifications to be made using text except for when they were driving (Figure 6.3). Almost all users wanted their notifications to be voice-based whilst they were driving. Seven percent of the users wanted all of their notifications to be made in text regardless of the context of use (Figure 6.4). The high percentage of text notifications implies that voice notifications should only be used in very specialised circumstances.

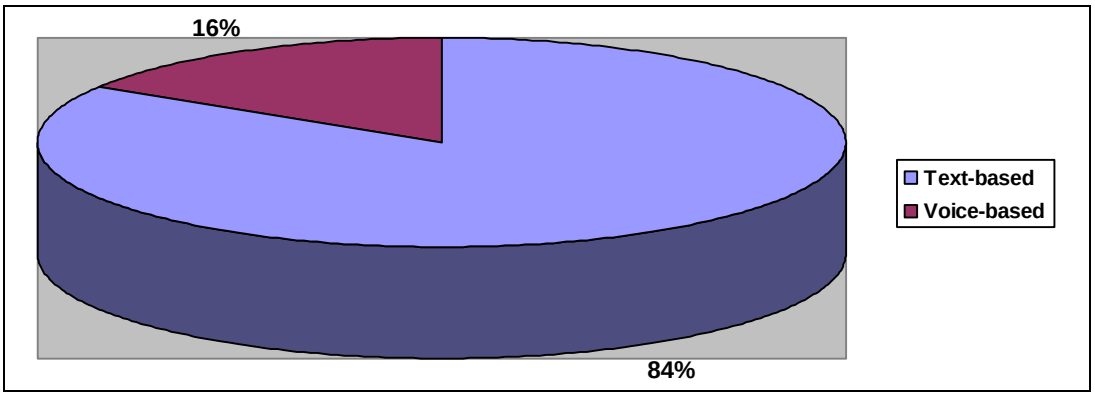

Figure 6.3: Mode preferences in all contexts except driving (N = 74)

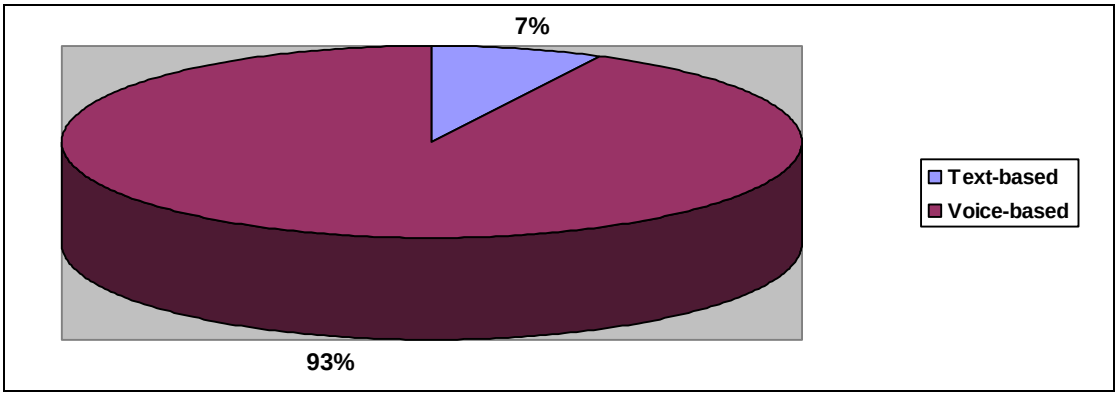

Figure 6.4: Mode preference while driving (N = 74)

During the first two weeks of training, once a notification was made the user was asked whether or not the notification was made in the correct mode. This feedback was used to further train the Preference Engine into deciding which mode the user actually wanted to be notified in, based on the user's context. An incorrect mode of notification could mean that either the user was unsure of the mode in which he/she wanted to be notified when entering his/her initial preferences, or that the incorrect context was determined. In order to prevent incorrect deductions, the user was presented with the context in which he/she were determined to be as well as three options which asked if the notification was in the correct mode, namely:

1. Yes;
2. no; and
3. wrong context

This allowed the users to inform the system that the incorrect context was determined as opposed to merely an incorrect choice in mode. This feedback facilitated conclusions about how accurate the Context Module was at determining the user's context and refinements to the weightings to achieve a more accurate context engine. The prototype performed 8,573 notifications over the two month evaluation period. Of the notifications that were made, 21.45% were made with the incorrect context being determined (Table 6.1). The weightings could be adjusted to facilitate more accurate context determination, but 78.55 % is a high enough percentage to assume that the correct context was generally determined.

	Text	Voice	Total
Correct Context Determined	66%	13%	**79%**
Incorrect Context Determined	10%	11%	**21%**
Total	**76%**	**24%**	**100%**

Table 6.1: Prototype notification summary (N = 8,573)

Based on the feedback provided by the users during the notification process, it was possible to determine whether or not the correct context and mode were chosen for each notification. This information allowed for amendments to the Context Module to ensure that the correct context was determined from the various inputs. It also allowed for fine tuning of each user's notification preferences. This would have

occurred when the users initially chose preferences without sufficient thought to what would be the most appropriate mode of communication for that particular context. Users may also have had an incorrect understanding of what the context meant.

Comparing the initial preferences chosen to the preferences learnt by the neural network provided an indication of how useful the initial preferences were. The preferences that were learnt were more accurate than the initial preferences since they were the preferences that the user chose whilst actually using the prototype. If the difference between the chosen preferences and the learnt preferences was consistently low (less than ten percent), we can conclude that the initial preferences are sufficient and asking the user if the notification was in the correct mode may prove more of a nuisance than a benefit. In practice, however, in all contexts (except driving) the users taught the prototype a different set of preferences from what they initially chose (Table 6.2). The percentage of users who changed their initial selection of modes was found to be over 50% for all contexts except driving. This shows that the users were either unsure of what the different contexts meant or they were unsure of the mode in which they wanted to be notified.

Context	Unchanged	Changed
At Desk	43.24%	56.76 %
In Meeting	1.81%	89.19 %
Sleeping	20.27%	79.73 %
Home (Other)	29.73%	70.27 %
Work (Other)	43.24%	56.76 %
Driving	72.97%	27.03 %

Table 6.2: Comparison between initial and learned preferences (N = 74)

Barkhuus and Dey provided research into the levels of usefulness and intrusiveness of context-aware mobile services (Barkhuus *et al.* 2003). They report that users are concerned with the lack of control in context-aware applications, the interruptability of the notifications and the sharing of sensitive contextual information. The sample user group was presented with a user satisfaction questionnaire every three weeks in order to gauge the effect of these concerns. Each question had a 10 point Likert scale

as an answer, ranking the user's concern from *None (1)* to *Highly concerned (10)* (Appendix A).

The questionnaires revealed that as the field testing progressed, the control and privacy concerns become less pronounced while concerns about interruptability increased (Table 6.3). There was a marked decrease in concerns regarding privacy and control which may indicate that, as the users become more familiar with the system, they began to trust it. The study also revealed an increase in concerns about interruptability as the study progressed (Figure 6.5). This may indicate that there was a certain degree of novelty around the prototype and once it wore off, the notifications became more of an annoyance than a help. This could also mean that the notifications were not filtered correctly and the users were receiving notifications which were of little use to them.

Evaluation	Control		Interruptability		Privacy	
Period	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
First	6.8	1.2	2.9	0.6	7.4	0.9
Second	5.7	1.3	3.8	0.9	5.2	0.5
Third	3.2	1.2	5.7	0.8	2.8	0.3

Table 6.3: Concerns from the user satisfcation questionnaire (N = 74)

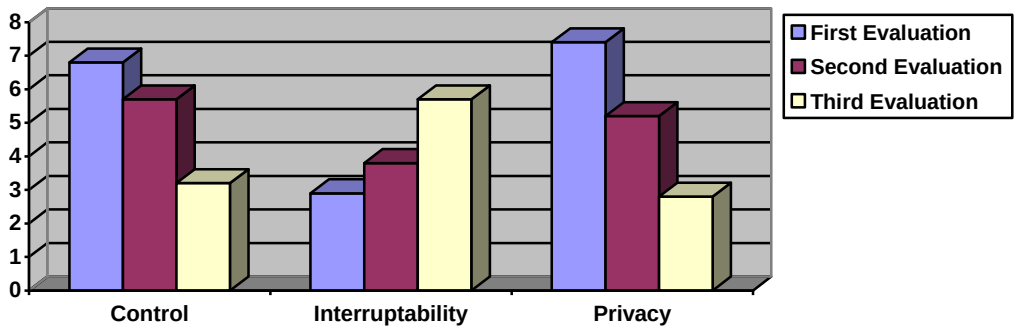

Figure 6.5: Concern trends from user satisfaction questionnaire (N = 74)

6.4 Extensibility Evaluation

During the initial prototype implementation, only one device type was catered for, namely cell phones. This was done so that the extensibility of the system could be shown by adding additional functionality in the form of supporting new devices and notification services.

The proposed model was designed to allow for new devices to be added by means of using various gateways and plug-ins. New services can be added by creating a new source which pushes notifications to the Trigger Management Service. This service is wrapped in a DLL and placed in the Notification Source folder. New devices can be supported by implementing the required input and output gateways as well as any widgets that could be used to determine the user's context.

In order to show how the proposed model could be extended, it was decided to add support for laptops and WiFi-enabled PDAs as devices. The additional service notified users of network state change for certain network enabled devices. Network changes were obtained from a network monitoring tool from Microsoft (Microsoft MOM) in order to determine what state the network was in.

In order to add support for the new set of devices, it was decided that the Internet would be used for input and output. The reason for this was that by using the Internet a laptop or PDA can be used as long as it has Internet connectivity regardless of the provider of the connectivity. This means that it is not limited only to WiFi networks, but extends to any network which supports Internet access.

Text input was implemented the same way as for the cell phones – by using the Internet and a browser on the device. The input screens for textual input on laptops and PDAs are the same ones created in .Net Mobile for the cell phones with the only difference being that a style sheet was used in order to make the forms look nicer on a large screen device.

Voice input could not be performed the same way for laptops as for cell phones. Laptops are not deemed as phone devices even though they do generally have built-in

modems and are capable of making phone calls. In order to support voice input it was decided that speech recognition software would be used to turn the spoken text into written text and send the written text to the Input Module using the Internet. This was implemented in a Windows Service which ran silently until the person held down the ALT and CTRL keys. This signalled the service to start decoding all voice input and send it to the Input Module.

The various pieces of input are sent to the Input Module which is responsible for converting the input into commands. This ensures that each command can be issued in various modes and from various sources with the only requirement being that the input sent makes sense in terms of a command for the notification system to understand in its current state.

Text-based notification for laptops and PDAs was implemented using a service which runs on the device. This service listens for notifications from the prototype and notifies the user with a pop-up message once it receives a notification. The service listens on a port which is open to the Internet. There were no firewall or other security restrictions in the test environment but if there were, a proxy service would have to be implemented in order to allow notifications to get past the security requirements.

Voice output was implemented on the same service as the textual notifications. The same service listened for notifications (both text and voice) and displayed or played the message based on the output mode. Voice messages are sent to the service in text form and are converted to voice using the Microsoft Speech SDK.

The contextual information between various devices is different. For example, a laptop does not have any standard means of determining if the user is busy with a phone call or not. Laptops and PDAs have access to location-based information as long as they have either GPS functionality or are within a mapped WiFi area. For the prototype, the laptops and PDAs were located using a mapped WiFi area. If the laptop or PDA was outside of WiFi range, location was not used as a source of contextual information. Each of the sensors was implemented as services which ran on the device except for location awareness. Location awareness was implemented

using a server side application which tracked the location of any device connected to a WiFi cloud.

The additional service that was added monitors network changes and notifies network administrators if the state of these devices changes. In order to monitor the network state of the devices, a Windows service was used which was responsible for periodically checking each of the network devices. If a device's state changed, an entry was made in a database table. This table was checked by a module added to the Trigger Management Module. If a new entry was detected, a device's state was changed and a notification was sent.

6.5 Conclusions

The prototype, called Mercury, was deployed in a live environment for a period of two months. During these two months a number of measures were used to gauge the usability and usefulness of Mercury.

The proposed model was evaluated using the same criteria used in Chapter 3 to evaluate existing models. This evaluation showed that the proposed model successfully addresses the limitations of the IBM INS model in terms of adaptive multimodal interaction (Section 6.2).

Extensive field testing was conducted over a period of three months to improve the notification preferences and evaluate the usefulness of Mercury (Section 6.3). This field testing showed that the users' initial preferences were often wrong and that they learnt preferences were more accurate.

The extensibility of the proposed model was evaluated by adding support for laptops and WiFi-enabled PDAs to Mercury (Section 6.4). The extensibility was evaluated further by adding an additional notification service to notify users of a change in the state of network devices. This evaluation showed that additional devices and services could be added to an application based on the proposed model without having to change the model.

The next chapter concludes the dissertation, summarising its contribution and achievements as well as outlining possibilities for future research.

Chapter 7 Conclusions

7.1 Introduction

The aim of this dissertation was to develop a model that would provide support for multimodal mobile notifications which adapt depending on the user's context. This model was implemented as a prototypr named Mercury in order to demonstrate its effectiveness. In addition, a network state monitoring tool was developed. This tool was then added to Mercury to determine how easily the model could be applied to new services. The prototype was then evaluated by means of field testing in order to determine how efficiently and effectively the model supported adaptive mobile notifications.

7.2 Research Achievements

This research resulted in both theoretical and practical achievements. Theoretical research achievements consist of a set of requirements that mobile notification services should satisfy (Section 7.2.1), the requirements that adaptive multimodal mobile notification services should satisfy (Section 7.2.2), the proposal of a model for adaptive multimodal mobile notification services (Section 7.2.3) and the results of a field study conducted over a period of three months (Section 7.2.6). Practical research achievements are the analysis of existing models for adaptive multimodal mobile notification services (Section 7.2.4) and the development of a prototype based on the proposed model (Section 7.2.5).

7.2.1 Requirements for Mobile Notification Services

The literature study of mobile notification services (Chapter 2) revealed that no comprehensive set of requirements exists that mobile notification services must satisfy. The identification of several common characteristics and user environment requirements has led to the development of a list of requirements a mobile notification service (Section 2.5.2). These requirements address four areas of mobile notification

services, namely the mobility of the users, the varied content formats, the large number of mobile users and the need for security and privacy.

7.2.2 Requirements for Adaptive Multimodal Mobile Notification Services

Adaptive multimodal mobile notification services need to satisfy additional requirements. These requirements are based on the requirements for a mobile notification service system and included various other factors applicable to adaptive multimodal mobile notification systems, namely context abstraction from context use, device and sensor flexibility, contextual history, distributed communication, time synchronisation and mode functionality independence (Section 3.4).

7.2.3 Development of a Model for Adaptive Multimodal Mobile Notification Services

Another achievement of this research was the proposal of a model based on the IBM INS model (Section 3.5.3). The proposed model extends the INS model by including contextual history, time based context and multimodal input. The proposed model is repeated in Figure 7.1 for ease of reference. The model is separated into an Input Module, an Output Module, a Trigger Management Service and a Context Module.

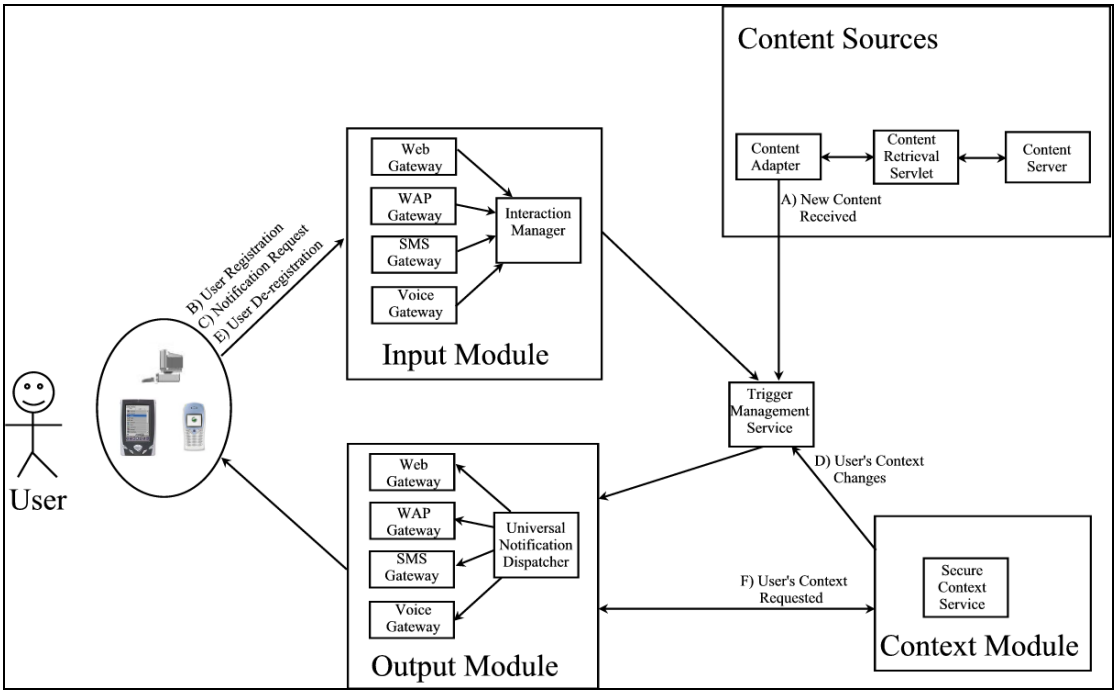

Figure 7.1: Proposed model for adaptive multimodal mobile notification service

The Input Module accepts user input from multiple modes and devices and parses the input into commands for the Trigger Management Service. All input sources have their own Recognition and Interpreter components. These components take the input from the source and convert the input into a format the system can understand. This format is independent of the mode or device from which the input originated. Once the input has been interpreted, the Integrator component combines the inputs from the various modes and devices into a single command for the system to use.

The Output Module accepts notification requests from the Trigger Management Service and sends the notification to the user based on the user's context. Once the Output Module receives a request for a notification it makes a request for the user's current context to the Context Module. The Context Module then requests information from the user's various widgets on his/her mobile device. Once the Context Module receives a result, combines the widgets' results into a single context and returns the result to the Output Module.

The Trigger Management Service is responsible for initiating notifications. The service continuously monitors the content sources for updated or new data. Once new content is found, the Trigger Management Service places a request for a notification

on the Output Module's Notification Request Queue. The Output Module then checks if the request warrants a notification based on the user's preferences.

7.2.4 Analysis of Existing Models for Adaptive Multimodal Mobile Notifications

A mobile notification system is considered an adaptive multimodal mobile notification system if it uses the user's context to possibly change the mode used for notifications.

Research into existing models for adaptive multimodal mobile notification revealed that no models exist that satisfy all the requirements in Section 3.4. The research revealed several shortcomings of the existing models:

- Existing models cater for multiple output modes but none cater for multimodality on both the input and the output;
- no existing models allow for contextual history to be used as another context source;
- once the mode has been chosen, it cannot be changed mid-notification; and
- there is no model which allows for contextual information to come from multiple separate locations – all sensors must occupy the same device.

7.2.5 The Implementation of an Adaptive Multimodal Mobile Notification Service

Another achievement of this research was the development of a prototype (called Mercury) based on the proposed model for an adaptive multimodal mobile notification service. The implementation showed the effectiveness of the proposed model for adaptive multimodal mobile notifications. The implementation aimed to show how an adaptive multimodal mobile notification service could be implemented.

The extensibility of the model was shown by adding an additional service to Mercury and by adding support for more devices. Mercury initially provided support for cell phones only, so laptops and WiFi-enabled PDAs were included at a later stage. Further extensibility was shown by means of adding a Network State Monitoring tool as an additional service which users could subscribe to. The Network State

Monitoring tool notified network administrators of changes of network states of critical network devices.

7.2.6 Field Testing Results

Mercury was used in a field test spanning a period of three months. The evaluation of Mercury yielded various theoretical results. The first of which being that whilst the users were using the prototype, the majority of them changed their preferred modes of communication for almost all contexts. This implies that the users were either uncertain of the most suitable mode, or they were unsure of what the context meant. Another important result was that after the initial month of use, the users started to find the notifications more intrusive and less helpful. This shows that either the notification services were not critical enough or that another approach needs to be used to determine whether a message should result in a notification or not. The most preferred mode of notification was found to be Text in almost all contexts. Voice was preferred only whilst the user was driving (Section 6.3.2).

7.3 Recommendations

7.3.1 Recommendations for Theory

The literature study revealed that there is a lack of uniformity with regard to methods of determining context. Each model and framework has a unique method of determining the context of a user. In many cases, the models were restricted to a specific set of device types as the methods the models use to determine a user's context require certain information which may not be available on all devices (Section 2.7).

There is also a lack of a uniform way for evaluating and comparing models against each other (Section 2.5.2). This research identified a set of requirements for an adaptive mobile notification service but there is still a lack of a uniform way of comparing models against each other. If two models both satisfy all of the requirements identified, there is still a chance that one of the models may be better suited to a particular environment than another.

7.3.2 Recommendations for Practice

Both the proposed model and Mercury could be used to develop adaptive multimodal mobile notification services as Mercury provides a way for developers to easily add additional notification services to an existing framework. This allows the developers to leverage the existing code-base and user preference library to make it quicker to add additional services. Mercury provides multimodal input and output as well as a Trigger Management Service. This means that a developer only has to create a new content source in order for a new service to be added.

7.3.3 Recommendations for Future Research

The proposed model and the prototype, Mercury, can both be improved with more research and development, as discussed below.

7.3.3.1 Determining Context

There is a lack of uniformity in determining a user's context and each model or framework typically uses a unique method for determining context. This limits the ability for models and frameworks to be integrated. The models could be aided by researching uniform, extensible methods for determining a user's context. This research could be further extended by researching a complete set of context's that a user could be in and the appropriate notification modes.

7.3.3.2 Research model comparison

Comparing models for adaptive mobile notification services is also undefined. A set of requirements for an adaptive mobile notification system was outlined in Section 2.5.2, but no formal methodology for comparing models and frameworks is yet available.

7.3.3.3 Implement support for more devices

Mercury currently only supports a limited set of mobile devices, namely:

- Cell phones;
- laptops;
- PDAs; and
- desktop PCs.

Support for additional devices can be added by implementing the required gateways, recognition and interpreter components. This functionality can be added to Mercury in order to extend the access of these services.

7.4 Summary

The goal of this research was to develop a model for an adaptive mobile notification service which uses context awareness to adapt to each user's environment and make mobile notification in the most suitable mode for each user. The goal was successfully achieved, both theoretically with the proposal of an extensible model for adaptive mobile notification using context awareness (Chapter 4), and practically with the successful implementation of Mercury as a prototype of the proposed model.

An in-depth literature study was conducted into existing adaptive multimodal mobile notification systems and their corresponding models (Chapter 2 and Chapter 3). It was found that there is no support for complete multimodality for input and output as well as no uniform way of evaluating models for adaptive mobile notification systems (Section 2.5.2 and Section 3.4). In order to address these shortcomings, a set of requirements was created to be used for evaluating existing models. These requirements were used in Section 3.6 to compare existing models and the IBM INS model was chosen as the most appropriate model and adapted so that it satisfied all the requirements (Chapter 4).

A prototype of the proposed model, called Mercury, was implemented to notify users of new email. The implementation evaluated the effectiveness with which the model could be implemented. Chapter 6 investigated the usefulness of adaptive multimodal mobile notification by evaluating Mercury in a field test. The field test showed that

concerns of interruptability of the notifications increased as time progressed. This suggests that the mobile notifications were initially useful but another approach needs to be used when filtering the messages which result in a notification. This research can therefore be regarded as successfully meeting its objectives and producing a contribution to the theory and practice of adaptive multimodal mobile notification.

Chapter 8 References

Abowd, G. C., Dey A. K., Brown, P. J., Davies, N., Smith, M. and Steggles, P., "Towards a better understanding of context and context-awareness", Handheld and Ubiquitous Computing, Springer, Berlin, pp. 304-307, 1999.

Afonso, A. P., Silva, M. J., "Dynamic Information Dissemination to Mobile Users," Mobile Networks and Applications, Volume 9, Issue 5, pp. 529-536, October 2004.

Asaf, A., Botzer, D., Etzion, O., Yatzkar-Haham, T., "Push Technology Personalization Through Event Correlation," Proceedings of the 26th International Conference on Very Large Data Bases, pp. 643-645, 2000.

Bazinette, V., Cohen, N. H., Ebling, M. R., Hunt, G. D. H., Lei, H., Purakayastha, A., Stewart, G., Wong, L., Yeh, D. L., "An Intelligent Notification System," IBM Research Report RC 22089, June 2001.

Black, B. and Larsson, E.A., "Beyond Simple Email - Upgrading an Entire Campus to Enterprise Email and Calendaring with GroupWise", SIGUCCS '06, November 5-8, 2006, Edmonton, Alberta, Canada, 2006.

Blackberry, Research in Motion (RIM) Demonstration Web site. http://www.blackberry.net/solutions/demo, Last Accessed 13 December 2007.

Blattner, M., M., Sumikawa, D., A., Greenberg, R., M., "Earcons and Icons: Their Structure and Common Design Principles," Human Computer Interaction, pp. 11-44, 1989.

Bluetooth Special Interest Group, "Bluetooth Protocol Architecture," 1999.

Bly, S., "Sound and computer information presentation", (Unpublished PhD Thesis UCRL53282), Lawrence Livermore National Laboratory, 1982.

Bolt, R., “Put-that-there: Voice and Gesture at the Graphics Interface”, Proceedings of the 7th International Conference on Computer Graphics and Interactive Techniques, International Conference on Computer Graphics and Interactive Techniques, Seattle, Washington, pp 262-270, 1980.

Brewster, S., A., “Chapter 12: Non-speech Auditory Output”, In Jacko, J. and Sears, A., The Human Computer Interaction Handbook, Lawrence Erlbaum Associates, pp. 220-239, 2002.

Brewster, S., “Overcoming the Lack of Screen Space on Mobile Computers”, Personal and Ubiquitous Computing, Volume 6, pp. 188-205, 2002.

Brewster, S.A. “Multimodal interaction and proactive computing”, in British Council workshop on Proactive Computing (Nizhny Novgorod, Russia), 2005.

Brown, P. J., Burleston, W., Lamming, M., Rahlff, O., Romano, G., Scholz, J. and Snowdon, D., “Context-awareness: Some Compelling Applications”, International Symposium on Handheld and Ubiquitous Computing (HUC ’00), 2000.

Buxton, W., Gaver, W., and Bly, S., “Tutorial Number 8: The use of non-speech audio at the interface”, Proceedings of the ACM CHI, New Orleans, 1991.

Castillo, E., Guijarro-Berdinas, B., Fontenla-Romero, O., Alonso-Betanzos, A., “A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis”, Journal of Machine Learning Research 7, pp. 1159-1182, July 2006.

Cellular-News, “List of countries that ban cellphone use while driving”, Available: http://www.cellular-news.com/car_bans/, Accessed 14 July, 2005.

Chen, G., and Kotz, D., “A Survey of Context-Aware Mobile Computing Research”, Technical Report: TR2000-381, 2000.

Cheriton, D., “Dissemination-oriented communication systems,” Technical Report, Stanford University, 1992.

Chozam, “Bluetooth Technology Information,” (2005). web site: http://www.chozamtech.com/communications/wirelesscomm/standards/bluetooth_technology.html, Last Accessed 01 June 2005.

Cohen, P.R., McGee, D.R., and Clow, J. (2000) "The efficiency of multimodal interaction." Demo for the Conference on Human Factors in Computing Systems (CHI’00) Extended Abstracts, The Hague, The Netherlands, 1-6 April, pp. 26-27.

Colby, J., “Multimodality: The Next Wave of Mobile Interaction”, Speech Technology, Volume 7, Number 6, pp 32, December 2002.

Coles, A. and Deliot, E. and Melamed, T. and Lansard, K. “A framework for coordinated multi-modal browsing with multiple clients”. In Proceedings International WWW Conference, pp. 718-726, 2003.

Costello, R., “Mobile Technology in the Real Estate Industry,” Global Real Estate Now, *Fall 2001*.

Cugola, G., Jacobsen, H. A., “Using Publish/Subscribe Middleware for Mobile Systems,” Mobile Computing and Communications Review, Volume 6, Number 4, pp. 25-33, 2002.

Decina, M. and Trecordi, V. "Convergence of telecommunications and computing to networking models for integrated services and applications",' Proceedings of the IEEE, vol. 85, no. 12, pp. 1887-1914, December 1997.

Dey, A. K., Salber, D., Abowd, G. D., “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications”, Human-Computer Interaction, 16, 2001.

Djuknic, G. and Richton, R. "Geolocation and Assisted-GPS," Lucent Technologies white paper, web site: http://www.lucent.com/knowledge/documentdetail/0.1983.inContentId+090094038000e51finLocaleId+1.00.html, 2003, Last Accessed 14 March 2005.

Du Toit, C., Van Der Walt, A., “Temporal Grammars”, Proceedings of SAICSIT 2002, pp 205-211, 2002.

Duh, H.B.L, Tan, G.C.B, Chen, V.H.H, “Usability Evaluation for Mobile Devices - A Comparison of Laboratory and Field Tests”, MobileHCI ’06, 12-15 September, Helsinki, Finland, 2006.

Dunnavant, S., “Blackberries in Support of Technology,” Proceedings of the 29th annual ACM SIGUCCS conference on user services, Oregon, USA, pp. 36-39, 2001.

Ebling, M. R., Hunt, G. D. H., Lei, H., “Issues for Context Services for Pervasive Computing”, Proceedings of the Advanced Workshop on Middleware for Mobile Computing, 2001.

FCC, "Enhanced 911," Federal Communications Commission, web site: http://www.fcc.gove/911/enhanced, 2003, Last Accessed 26 February 2005.

Gaver, W. W., “The SonicFinder: An interface that uses auditory icons,” Human Computer Interaction, pp. 67-94, 1989.

Ghini, V., Pau, G., Salomoni, P., “Integrating notification services in computer network and mobile telephony,” Proceedings of the 2000 ACM symposium on Applied computing, pp. 549-553, 2000.

Gopal, R.D., Walter, Z., Tripathi, A.K., “Admediation - New Horizons in Effective Email Advertising”, Communications of the ACM, December 2001/Vol. 44 No. 12, pp. 91-96, 2001.

GSM World, "What is General Packet Radio Service," (2005), web site: http://www.gsmworld.com/technology/gprs/intro.shtml, Last Accessed 01 June 2005.

Henricksen, K. and Indulska, J., "A Software Engineering Framework for Context-Aware Pervasive Computing", Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications, pp. 77-86, 2004.

Hidding, J. G., "Reinventing Methodology - Who Reads it and Why", Communications of the ACM No. 11 Vol. 44, pp. 102-109, 2007.

IEEE, "The IEEE 802.16 Working Group on Broadband Wireless Access Standards", web site: http://ieee802.org/16/, Last Accessed 01 June 2005.

ITU (2005) "ICT free statistics", International Telecommunications Union. Available: http://www.itu.int/ITU-D/ict/statistics, 2005, Last Accessed 30 May, 2005.

Jöst, M., Häußler, J., Merdes, M., and Malaka, R., "Multimodal Interaction for Pedestrians: An Evaluation Study," Proceedings on the 10th international conference on Intelligent user interfaces, San Diego, California, USA, pp. 59-66, 2005.

Kaasinen, E., "User needs for location-aware mobile services", Personal and ubiquitous computing, volume 7, issue 1, pp. 70-79, May 2003.

Keshav, S., "Why Cell Phones Will Dominate the Future Internet", ACM SIGCOMM Computer Communication Review, Volume 35, Number 2, April, pp. 83-86, 2000.

Lamming, M., Brown, P. J., Carter, K., Eldridge, M., Flynn, M., Robinson, P. and Sellen, A., "The design of a human memory prosthesis", Computing Journal 37 (3), pp. 153-163, 1994.

Larson, J., A. and Raman, T., V., "W3C multimodal interaction framework." http://www.w3.org/TR/mmi-framework, W3C Note, May 2003, Last Accessed: 13 December 2007.

Love, S. and Perry, M., “Dealing with Mobile Conversations in Public Places - Some implications for the design of socially intrusive technologies”, CHI 2004, 24-29 April, Vienna, Austria, pp. 1195-1198, 2004.

Marcus, A., “Mobile User-Interface Design for Work, Home, Play and On the Way,” , HCI 2004 Tutorial Proposal, 23 May 2005.

Marti, S., “How does the user interface design of mobile devices influence the social impact of mobile communication,” Qualifying Exam, MIT Media Lab, 2002.

McGee, D.R., and Cohen, P.R., "Exploring Handheld, Agent-based Multimodal Collaboration." Presented at the Workshop on Handheld Collaboration at the Conference on Computer Supported Cooperative Work (CSCW’98), Seattle, WA, Nov. 14-18, 1998.

Microsoft, Microsoft Outlook 2007, 2007.

Minch, R. P., “Privacy Issues in Location-Aware Mobile Devices,” Proceedings of the 37th annual Hawaii international conference on Systems Sciences (HICSS ’04) Track 5 – Volume 5, pp. 50127.2, 2004.

Mitchell, K., “A Survey of Context-Awareness”, Available: http://www.comp.lancs.ac.uk/~km/papers/ContextAwarenessSurvey.pdf, 2002, Last Accessed: March 24, 2005.

Munson, J. P. and Gupta, V. K., "Location-Based Notification as a General-Purpose Service," WMC, 2002.

Nah, F., Siau, K. and Sheng, J. “The Value of Mobile Applications: A Utility Company Study”, Communications of the ACM, Volume 48, Issue 2, pp. 85-90, February 2005.

Neufeld, D. J., Dyck, B., and Brotheridge, C. M., “An Exploratory Study of Electronic Mail as a Rich Communication Medium in a Global Virtual Organization,”

34th Hawaiian International Conference on System Sciences, Maui, HI, pp. 8021, 2001.

Nigay, L. and Coutaz, J., “A Design Space For Multimodal Systems: Concurrent Processing and Data Fusion”, Proceedings of the SIGCHI conference on human factors in computing systems, pp. 172-178, 1993.

Niklfeld, G., Finan, R., Pucher, M., “Multimodal Interface Architecture for Mobile Data Services”, Workshop on Wearable Computing 10-2001, Graz, Austria, 2001.

Nord, J., Synnes, K. and Parnes, P. "An Architecture for Location Aware Applications," Proceedings of the 35th Hawaii International Conference on Systems Sciences, pp. 293, 2002.

Oh, L. B. and Xu, H. "Effects of Multimedia on Mobile Consumer Behaviour: An Empirical Study of Location-Aware Advertising", 24th International Conference on Information Systems, Seattle, pp. 679-691, 2003.

Oviatt, S., “Taming recognition errors with a multimodal interface”, Communications of the ACM, Volume 43, Number 9, pp. 45-51, 2000.

Oviatt, S., “Multimodal System Processing in Mobile Environments”, Proceedings of the 13th Annual ACM Symposium on User Interface Software Technology, pp. 21-30, 2000.

Oviatt, S, and Cohen, P., “Multimodal interfaces that process what comes naturally”, Communications of the ACM, Volume 43, Number 3, pp. 45-53, 2000.

Oviatt, S., “Multimodal Interfaces”, Chapter in Handbook of Human-Computer Interaction, J. Jacko & A. Sears, pp. 286-304, 2002.

Oviatt, S. “Breaking the robustness barrier: Recent progress on the design of robust multimodal systems”, Advances in Computers. M. Zelkowitz, Ed. Academic Press, pp. 305-341, 2002.

Patterson, C. A., Muntz, R. R., Pancake, C. M., “Challenges in Location-Aware Computing,” Pervasive Computing, IEEE, Volume 2, Issue 2, pp. 80-89, 2003.

Preece, J., Rogers, Y. and Sharp, H., “Interaction Design : beyond human-computer interaction”, John Wiley & Sons, Inc, 2002.

Roth, J., “Patterns of Mobile Interaction,” Personal and Ubiquitous Computing, Volume 6, Issue 4 (September 2002), 2002.

Ruuska-Kalliokulju, S., Schneider-Hufschmidt, M., Vaananen-Vainio-Mattila, K. and Von Niman, B. Shaping the future of mobile devices - results of the CHI2000 Workshop on: "Future Mobile Device User Interfaces". Proceedings of Mobile HCI 2001: 3rd International Workshop on Human Computer Interaction with Mobile Devices, at IHM-HCI 2001, Lille, France, pp. 368, 10 September 2001.

Sawhney, N., Schmandt, C., “Nomadic Radio: Speech and Audio Interaction for Contextual Messaging in Nomadic Environments”, ACM Transactions on Computer-Human Interaction, Volume 7, Number 3, pp. 353-383, September 2000.

Schilit, B., Adams, N., Want, R., “Context-aware computing applications”, Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, pp. 89-101, 1994.

Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. V. and Velde, W. V., “Advanced Interaction in Context”, Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC ’99), pp. 89-101, Karlsruhe, Germany, Springer-Verlag. September 27-29, 1999a.

Schmidt, A., Beigl, M., Gellersen, H. W., “There is more to Context than Location”, Computers and Graphics, Vol. 23, No. 6, pp 893-901, 1999b.

Schmidt, A., Holleis, P., Hakkila, J., Rukzio, E., Atterer, R., “Mobile Phones as Tool to Increase Communication and Location Awareness of Users”, Mobility ’06, Oct 25-27, 2006, Bangkok, Thailand, 2006.

Scourias, J., “Overview of the Global System for Mobile Communications,” web site: http://ccnga.uwaterloo.ca/~jscouria/GSM/gsmreport.html, Last Accessed 19 June 2005.

Siewiorek, D., Smailagic, A., Furukawa, J., Moraveji, N., Reiger, K., and Shaffer, J., “SenSay: A Context-Aware Mobile Phone”. In Proceedings of IEEE International Symposium on Wearable Computers (ISWC), pp. 248-249, 2003.

Steiert, H.P., “Towards a Component-based n-Tier CS-Architecture”, ISAW3 Orlando Florida, USA, 1998.

Steinfield, C., "The Development of Location Based Services in Mobile Commerce", Michigan State University,. Available: http://www.msu.edu/~steinfie/elifelbschap.pdf, 2003, Last Accessed: 13 December 2007.

Vollmer, G., Gabner, K., “Quality Improvement of Email Communication in Work Groups and Organizations by Reflection”, Group ’05, November 6-9, 2005, Sanibel Island, Florida, USA, 2005.

W3C, “Voice eXtensible Markup Language (VoiceXML) version 1.0”, http://www.w3.org/TR/2000/NOTE-voicexml-20000505, 2000, Last Accessed: 13 December 2007.

Wei, B., Chen, Y., Huang, H., Jana, R., “A Multimedia Alerting and Notification Service for Mobile Users,” AT&T Labs – Research, web site: http://www.research.att.com/~rjana/MobEAII-Paper_3.pdf, Last Accessed 05 May 2005.

Winograd, T., “Architectures for Context”, Human-Computer Interaction, 16(2—4): pp. 401-419, 2001.

Wright, C., Chadha, K., “Beyond A-GPS: Multimode Location Technology,” Sirf, white paper, web site: http://www.sirf.com/downloads/collateral/Press_Releases/BeyondAGPS.PDF, Last Accessed 13 May 2005.

Yoshimura, M., Hamaguchi, N., Kozaki, T., Takatori, M. and Higaki, N., “High-performance and Data-optimized 3rd-generation Mobile Communication System: 1xEV-DO,” Hitachi Review, Vol. 53, No. 6, 2004.

Appendix A User Satisfaction Questionnaire

1. Introduction to Mercury

Mercury is an adaptive multimodal mobile notification service developed to send you notifications of information that is important to you. Your help during this evaluation process will help the development team make decisions about future releases of Mercury. Your feedback will be used to help us understand how Mercury needs to be adapted to best suit your needs.

2. Question Explanations

The questions are all user satisfaction questions. They are designed to provide insight into which parts of Mercury make you feel comfortable and which cause you concern, Each question has a rating from 1 (None, or Strongly Disagree) to 10 (Extremely Concerned, or Strongly Agree). Please select the most appropriate rating for the amount of concern you feel for the question.

3. Questions

Question											
Privacy (is your context being determined safely and accurately) *1 = None, 10 = Extremely Concerned*											
1	Is my privacy being invaded?	1	2	3	4	5	6	7	8	9	10
2	Is my contextual information used securely?	1	2	3	4	5	6	7	8	9	10
3	Is it my contextual information?	1	2	3	4	5	6	7	8	9	10
4	Is my contextual information accurate?	1	2	3	4	5	6	7	8	9	10
5	Is any contextual information used without my knowledge?	1	2	3	4	5	6	7	8	9	10
Interruptability (is Mercury helping you or being an annoyance) *1 = Strongly Disagree, 10 = Strongly Agree*											
1	I am getting too many notifications a day	1	2	3	4	5	6	7	8	9	10
2	My notifications are in the best mode	1	2	3	4	5	6	7	8	9	10

3	The notifications disturb me	1	2	3	4	5	6	7	8	9	10
4	Are only important notifications made?	1	2	3	4	5	6	7	8	9	10
5	The notifications contain enough information to be useful	1	2	3	4	5	6	7	8	9	10
Control (Do you have control of Mercury's notifications) *1 = Strongly Disagree, 10 = Strongly Agree*											
1	I have control over the notifications that are made	1	2	3	4	5	6	7	8	9	10
2	I have control over what contextual information Mercury has access to	1	2	3	4	5	6	7	8	9	10
3	I know what mode the notifications will be made in when I receive them	1	2	3	4	5	6	7	8	9	10
4	I can control the frequency my contextual information is accessed	1	2	3	4	5	6	7	8	9	10
5	I can control how detailed the notifications are	1	2	3	4	5	6	7	8	9	10