2006-01-01

Point of Care Healthcare Quality Control for Patients Using Mobile Devices

Owen Lynch
Technological University Dublin

Point of Care Healthcare Quality Control For Patients Using Mobile Devices

Owen Lynch BE

School of Control Systems and Electrical Engineering,

Dublin Institute of Technology

A thesis presented to Dublin Institute of Technology,

Faculty of Engineering

For the degree of

Master of Philosophy (MPhil)

January 2006

Research Supervisors:

Mr. John McGrory
Dr. Eugene Coyle
Dr. Mike Murphy

Abstract

The advances made in the domain of mobile telecommunications over the last decade offer great potential for developments in many areas. One such area that can benefit from mobile communications is telemedicine, which is the provision of medical assistance, in one form or another, to patients who are geographically separated from the healthcare provider. When a person is ill, individual attention from medical professionals is of the utmost importance until they have returned to full health. However, people who suffer with long term and chronic illnesses may need life long care and often must manage their condition at home. Many chronically ill patients manage their condition themselves and perform 'self-testing' with Point of Care Test (POCT) equipment as part of this condition management. When a specimen sample is analysed at home with a POCT device, a result is available to the patient almost immediately, but the result cannot be proven to be plausible for the patient unless it is validated by the hospital systems. In addition to this the hospital is unaware of the patients condition and progress between hospital visits. This research addresses some of the issues and problems that face patients who use POCT equipment to 'self-manage' their condition at home. Using mobile phone technologies and the Java platform, three alternative methods for providing patients with a service of POCT result validation and storage were designed. The implementation and test of these systems, proves that a mobile phone solution to the issues associated with patient self-testing is possible and can greatly contribute to improving the quality of patient care.

Declaration

I certify that this thesis which I now submit for examination for the award of *MPhil*, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate study by research of the Dublin Institute of Technology and has not been submitted in whole or in part for an award in any other Institute or University.

The work reported on in this thesis conforms to the principles and requirements of the Institute's guidelines for ethics in research.

The Institute has permission to keep, to lend or to copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

Signature: _____

Owen Lynch

Date: 9th January 2006

Acknowledgements

I would like to thank the following people without whose help and support I would have had great difficulty completing this research. Mr. John McGrory, my project supervisor, for his guidance and assistance over the course of the project and Dr. Eugene Coyle and Dr. Mike Murphy for their helpful criticism during the writing of this thesis. Maggie and my family for their constant support, advice and understanding. Also, I would like to thank my colleagues in the research department namely Will, Dan, Eileen, Jim, Mark, Niall, Pauline and Sharon for always being happy to discuss a problem even if they didn't know the solution.

Table of Contents

List of Figures

List of Tables

1 Introduction

This chapter presents the aims and scope of this MPhil research project. The problems addressed by the research will be explained to highlight its importance. A detailed discussion of the research area and technologies will be given in subsequent chapters.

1.1 Background

This research is in the area of telemedicine. Telemedicine uses telecommunications technology to connect patients with healthcare providers at a distance. For example, a nurse can transmit digital images of a patient's condition to a doctor, house alarms can alert medical services to an elderly patient's fall and home care systems can monitor the vital signs of the housebound. The older share of the population, with whom chronic illnesses are associated, is set to double by 2030 [1], which heightens the need for developments and innovation in telemedicine.

Patients with many different conditions, especially chronic illnesses like diabetes, are often required to self-manage their condition and perform Point of Care Testing (POCT) at home. Patient self-testing and POCT in the home is a rapidly growing area in the healthcare arena. It gives patients an opportunity to take control of their conditions and can reduce their length of stay in hospital. The results of these tests can be used to help determine medication dosage or simply to monitor their condition. However, this patient ‘self-monitoring’ raises quality control issues that would not arise in a clinical setting, since the sample acquisition and testing procedures are not overseen by professional hospital staff. In hospital the nurses and laboratory staff must be trained and certified in the testing procedure, the instrumentation used to perform the test, and quality control practices. There is no such requirement for consumers who purchase home tests, even the ones prescribed or recommended by their doctors.

Another key issue, the main focus of this research, is that results from such tests are not clinically validated to ensure that they are plausible for that patient at the time of testing. In hospital, tests taken by clinicians are validated by dedicated computer validation

systems as part of the quality control process, before a diagnosis is made. Thus, for people testing at home with POCT devices, there is a need to implement a system of result validation, either locally or by the hospital validation system itself.

Communications and mobile phone technologies are rapidly developing. Most mobile phones available today are more than simple voice-centric devices. They are complex data communication terminals and are continually evolving to an "always on" form of network computer, capable of availing of Internet services continuously. Java 2 Micro Edition (J2ME) comprises a set of technologies and specifications developed for small devices with limited memory and computation possibilities such as mobile phones. It encompasses a portable, platform independent language so applications developed with J2ME can run on any mobile device that supports Java.

1.2 Project Aim

This research takes advantage of the ubiquity of mobile phones and how they may be used to link patients, who manage their condition in the home, with the hospital information system (HIS) to upload and validate their results. The aim of this research is to investigate the use of mobile communications to allow patients using POCT equipment to link into the hospital sample validation systems so that their sample can be independently validated. In addition to validating the result, the result can be stored in the patient's electronic patient record (EPR). It is proposed that a generic Java application will be developed to run on a mobile phone, which will take the result value from the patient's POCT device, transmit it to the hospital for validation and storage and then return instructions or advice to the patient based on the outcome of the validation process.

Three different versions of this system have been designed and a proof of concept of each of these versions was implemented and tested. Each version consists of J2ME client applications, and more powerful server programs for processing the data and storing it in databases (to simulate the HIS and electronic patient record) [2]. The system proposed could help patients better manage their condition, ensure a higher level of quality control in the results they obtain and keep the hospital up to date with their medical status.

1.3 Thesis Outline

Following upon this introductory chapter, Chapter 2 will embrace the relevant medical topics that this research is concerned with. This includes telemedicine, hospital protocols and information systems, laboratory testing, point of care testing and patient self-testing.

Chapter 3 discusses the technical aspects of the thesis. This work is primarily concerned with mobile phone client applications, thus mobile phone capabilities and wireless networks are investigated. It also discusses client and server technologies relevant to the research and that were used in developing the proof of concept system.

In Chapter 4, the proposed solution to the problems that this research addresses is introduced. The systems to enable patients to transmit test result data to the hospital are discussed and the design of these systems specified.

The proof of concept applications for the three versions of the system were implemented and tested. A full discussion of all the client and server programs implemented is discussed in Chapter 5.

Chapter 6 concludes the research and comments on the outcomes of developed implementations. Some suggestion and ideas for improving the system in any future work are proposed.

Additional information including a nomenclature and publications are provided in the appendices. A CD containing all of the source code for the applications described is also included with this thesis.

2 Overview of Telemedicine and Medical Informatics

The purpose of this chapter is to give the reader some background information on telemedicine, medical informatics, and patient self-testing, three of the main areas that this research is concerned with. After an introduction to telemedicine various relevant sections of medical informatics will be discussed. This includes hospital data communication protocols, electronic patient records and clinical laboratory methods for testing and validating results. Point of care testing and patient self-testing will be examined as will the problems associated with home-based self-testing. Finally some point of care equipment will be explained together with information on the medical conditions that they are used to manage.

2.1 Telemedicine

The remote delivery of health care to patients is known by many terms including telecare, telehealth, e-health and telemedicine and these terms are often used interchangeably [3]. For simplicity this thesis will use telemedicine. Telemedicine is the use of information and communication technologies (ICT) to provide clinical care to individuals at a distance and the transmission of information to provide that care [4]. Telemedicine is mainly used to provide patients who are living in rural and remote locations with quality and specialty care. It is a broad field and can range from a simple case of two physicians discussing a patient over the phone to the high speed and high bandwidth transmission of commands to a surgical robot in another country [5]. Other applications of telemedicine include the use of video-conferencing to provide tele-consultation to a patient in a remote location and the exchange of data between health centres [6]. This data could be high-resolution images for a dermatologist, ECGs or elements of a patient record.

The upsurge in interest and use of telemedicine recently is mainly due to the advances that have been made in ICT since the end of the 20th century, but telemedicine has been around for thirty or so years and was first used by NASA during their space program in the 1960's to monitor the physiological parameters of astronauts [7]. Although the efficacy of Telemedicine is still an area of much research, it has been shown that home-

based telemedicine in the area of chronic disease management is the most efficient application [8] [9].

Telemedicine is also concerned with at-home patients who self-manage their conditions [10]. These are generally patients with chronic diseases such as diabetes, cardiovascular diseases, chronic respiratory diseases such as asthma, and some cancers as well as post-surgery patients. The benefits of moving the care of a patient from formal health care systems such as hospitals and clinics to the home are clear for both the patient and the health care system. It allows patients to become more active in their own care, can reduce the length of stay in hospital and gives patients more independence as they do not have to make as many regular trips to hospital. Patient self-testing is the most patient-friendly method for long-term condition management and can provide outcomes that are as good as traditional methods [9]. It is this type of patient, who self-manage their conditions with POCT, that this research is primarily focused on. Patient self-testing is discussed further in section 2.6.2.

2.2 Hospital and Laboratory Communication

Data in hospitals are handled by large information systems. These systems can be broken down to enterprise and department level. They are responsible for communication of data within the hospital and for storing information. The Hospital Information System (HIS) is responsible for all data handling at the enterprise level. Specific healthcare domains within the hospital, such as the laboratory, pharmacy or radiology department have their own sub-systems. These departmental systems are dedicated to the internal processes of the department and they also deal with device data. For instance, control of analytical equipment and collection of result data in the laboratory or collection and storage of images such as x-rays in the radiology department.

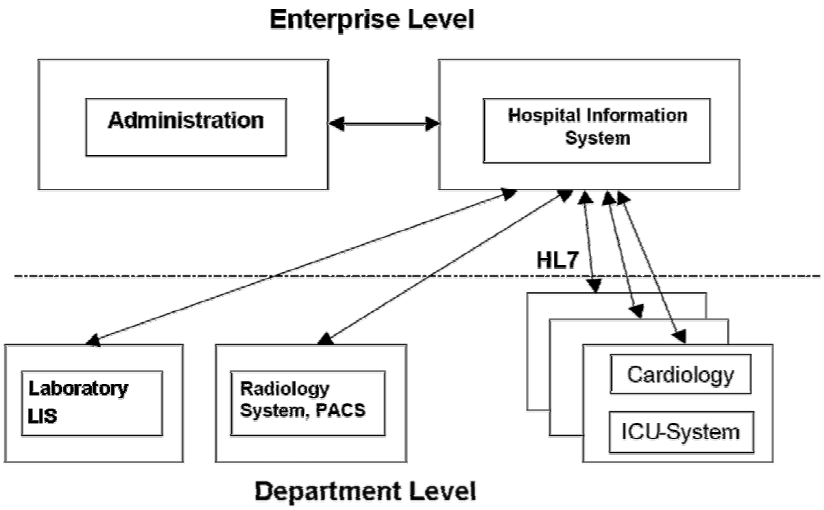

Figure 1: Departments and communication structure in hospitals [11] (modified)

In addition to this internal data processing, these systems also communicate with the hospital information system. For example, obtaining demographic data of a patient or storing results in a patient record [11]. Figure 1 depicts a simplified block diagram of the different enterprise and department systems in a hospital and how they communicate with each other. To enable these systems to communicate with each other, agreed messaging standards must be used. Health Level Seven (HL7) is the messaging standard most widely used for HIS communication and interfacing between hospitals, insurance companies and other health organisations [12]. HL7 is discussed further in section 2.4. As this research is concerned with laboratory systems and analyser results only, the Laboratory Information Systems (LIS) on the departmental level will be discussed further.

2.2.1 Hospital Information Systems

Data communication and processing at the enterprise level in a hospital are handled by the Hospital Information System (HIS). A hospital information systems purpose is to collect, store, process, retrieve and communicate patient and administrative information. So a comprehensive HIS is responsible not only for clinical information but for hospital admissions and discharge, billing and finance. In addition, the HIS is used to support clinical research, through use of the HIS database, and teaching. The users of the HIS are wide ranging and could be a nurse entering or retrieving patient data or the head of a

clinical department creating a service plan [13]. According to Van Bemmel et. al. [13] a HIS should at least contain the following four elements:

- A facility for the storage of data such as a database,
- facilities for data entry, retrieval and update from the database (i.e. a user interface application),
- data communication facilities (e.g. a Local Area Network), and
- facilities to allow users to use the system (i.e. computers or workstations)

2.2.2 Laboratory Information Systems

Clinical chemistry and Hematology departments in hospitals provide a very useful service to doctors and provide them with meaningful information on the chemical and cellular compositions of body fluids. With this information, doctors can confirm a suspected diagnosis, monitor effects of treatment and exclude or screen for the presence of disease. Modern laboratories consist of automated systems and analytical instruments. Laboratory Information Systems (LIS) are responsible for handling clinical laboratory data and processes. Some of the key tasks carried out by the LIS are test request processing, test ordering, sample labelling, specimen analysis, result validation, report generation and documentation. The LIS can interface with the HIS in order to obtain demographic data from the electronic patient record and to update information and test results in the record.

2.2.2.1 ASTM E1934

The American Society for Testing and Materials (ASTM) develops standards in a diverse range of industries [14]. ASTM E1394 is a standard that is very widely used in hospital laboratories. It specifies the message structure for communication between analytical instruments (AI) and a LIS and it allows almost any AI interface with almost any LIS [15]. Many analytical instruments today offer data-exchange facilities that use the ASTM E1936 standard. A message conforming to the standard consists of the following record types: Message Header, Patient Identifying Record, Test Order Request, Result, Comment, Request Information, Scientific Record and Manufacturer Information [12].

2.2.2.2 Integrated Networked Clinical Analyser (INCA) [16]

INCA is the work of a small consortium between Tallaght Hospital (AMNCH), the Dublin Institute of Technology and Trinity College Dublin. It defines a standard for the networking of clinical analyser instruments in hospitals and an interface between them and the Laboratory Information System. The INCA system was developed using Common Object Request Broker Architecture (CORBA) middleware, which is discussed in more detail in Chapter 3. This means that it is platform independent and it is designed to easily implement and merge with existing HIS's and LIS's. INCA provides a system to allow clinicians in a hospital request tests on a specimen sample of a patient from anywhere within the hospital. These requests are put in a work-list in the INCA system and the relevant analyser is used to perform the requested tests. Results are produced which undergo quality and validation checks. Finally they are placed into the patient's record and can be viewed by the doctor who ordered the tests. This is a typical laboratory process and is discussed in more detail in section 2.5. A Java version of the INCA system was implemented for this research and is described in Appendix C.

2.3 Electronic Patient Record (EPR)

Most hospitals in Ireland file patient data in a paper-based patient record. This is the traditional method for storing patient information and has been in use for many years. The paper-based record contains clinical notes relating to the patient that are often supplemented with data from various hospital departments such as laboratory test results, electrocardiograms and x-rays. The need to migrate from paper-based medical records to an electronic version has been recognised for many years now and as technology advances the drive is stronger. The many disadvantages of the paper-based record are another reason to move to an electronic form. These include [13]:

- A paper-based record can be lost, stolen or go missing.
- The text is hand written in the record and hence may be illegible or incomplete. This could give rise to misinterpretation of the data by another clinician.
- Paper-based records can only be in one place at a time.
- Paper-based records are difficult to back up and hence remain susceptible to physical damage due to fire or flooding and ageing.

- Searching for information within a record and across all records is a manual task and is very time consuming.
- Paper-based records can only contribute passively to the treatment of patients. Doctors do not have state-of-the-art knowledge at their fingertips and the paper record cannot draw a doctor's attention to abnormal laboratory results or contra-indications to drugs.

An electronic patient record (EPR) provides a secure, well-structured and formal method for storing patient information. As computers play a larger role in all aspects of hospital systems it makes more sense to use computer based medical records. For example, medical images from x-rays and MRI scans are now stored digitally. An EPR allows many different data types to be stored including text, audio, video and digital images and this data can be viewed by any doctor or clinician within the hospital [6]. Ideally, an EPR will store all clinical information about the patient from the "cradle to the grave" and will be common to both their General Practitioners (GPs) and hospitals. Such a record is being implemented by the National Health Service (NHS) in the UK, where a centralised "NHS care record" is being used. This is replacing the current system which consists of paper records and electronic systems in surgeries that cannot communicate with each other [17].

There are many advantages to using an electronic form of the patient record. An enormous quantity of data can be stored in an EPR, as storage media technology advances it is becoming cheaper to store more data. Hard disk drives today are capable of storing hundreds of Gigabytes. Information is entered to the record via forms, which ensure that all relevant data are entered in a legible and understandable manner and stored in the right location. The data are stored in a database so that complex queries specific to only certain attributes can be made and results retrieved easily and instantly. Databases allow multiple individuals to access the records simultaneously and as they are networked the users can be separated geographically so they are available to those who have access permission. The ability to easily search through all records can also help a researcher to identify trends (trending) and can help a hospital in their healthcare planning. An electronic record can be made secure and access to it can be restricted to prevent unauthorised users accessing them. The data can be encrypted so they are

unintelligible to anyone who does manage to hack into the system. As well as being able to store all of the static clinical data of a patient, an electronic record can be linked to intelligent systems to allow it play an active part in patient care. This will enable it to provide clinical data, suggest certain tests or procedures be carried out and automatically prescribe drugs for the patient. Alerts and reminders can be scheduled to help remind a doctor to carry out an action such as order certain tests on the patient.

The introduction of electronic records raises some ethical issues concerning patient rights and confidentiality. These include issues such as whether the patient should have the right to only allow selective access to their record and be able to conceal certain information from certain doctors as well as to be able to restrict casual distribution of the information to other healthcare providers [18]. These are important issues and must be considered when a health service introduces an electronic record system. However, they do not take away from the advantages discussed earlier and using an electronic patient record ultimately helps to provide greater care and benefits the patient greatly.

The applications that are to be discussed in this thesis require some form of an electronic patient record to be in place as the automated validation systems require patient specific data and the digital information from the patient must be stored. Hospital validation systems will be discussed later in the chapter.

2.4 Health Level Seven (HL7)

Health Level Seven is an American National Standards Institute (ANSI) accredited standard for hospital data communication. It is the international standard for all electronic data exchange in healthcare and can define, for example, the exchange of a message between the EPR and the laboratory system to order a test for a patient. It covers a wide range of medical messaging including patient admission / discharge, orders, test results, and billing [6]. It provides interoperability between all departmental systems in the hospital and is also commonly used for interfacing between hospitals, insurance companies and other health organisations. The term ‘Level 7’ refers to the application level in the OSI communications model. The application level defines the data to be exchanged and the timing of the exchange as well as supporting functions such as security checks and data exchange structuring [19].

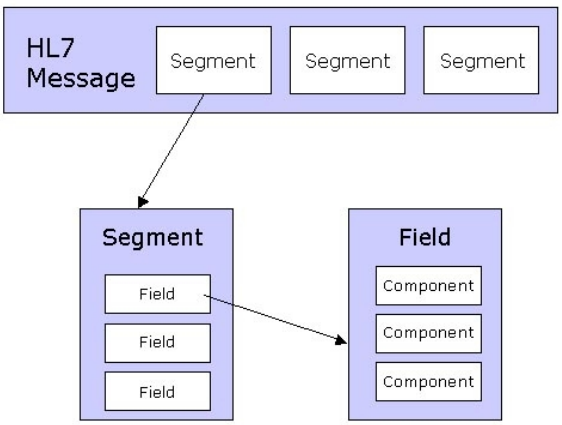

Figure 2: HL7 2.x message structure [6]

The current version of the messaging standard is HL7 Version 3.0. The older 2.x series of standards have continually evolved over the years and collectively represent the most widely implemented standard for healthcare information in the world [19].

The 2.x messages consist of a group of segments in a sequence, where a segment is a logical group of data fields, which can be broken down further if necessary into components [20]. Each segment contains information of a specific type and there are many segment types defined in the HL7 standard. For example, **MSH** is the code for a HL7 message header, which contains fields such as a time-stamp and sender information. **PID** is a patient identification segment, which includes patient demographic information. Another segment relevant to this thesis is the **OBX** segment. This is an observation / result segment and contains information such as the result type, result ID, units and date/time of result. HL7 version 2.x message structure is shown in Figure 2. Each data field is separated by a '|' character. An example of what the **PID** segment of a HL7 2.x message is shown in Listing 1.

```
PID|||A123456789||Smith^John||19551212|M|||24SomeStreet
^^Dublin25^IRELAND||(01)1234567|
```

Listing 1: HL7 2.x message

HL7 version 3 messages are written as XML (eXtensible Markup Language) format documents, which is the universal format for structured documents and data exchange on the Internet. XML is discussed in more detail in Chapter 3.

Each message is a string of text with information enclosed by tags, which are derived from the HL7 Reference Information Model (RIM). The RIM defines the grammar of HL7 messages and the basic building blocks of the language. An example of how the same **PID** segment shown above may look in HL7 version 3 format is shown in Listing 2. An XML HL7 document is a hierarchical structure with the segments, fields and components represented as XML elements. The segment **PID** is a second level element and the Field elements are third level elements. **<PID.11>** is the address Field, which has components **<XAD>**, the extended address data type.

```
<?xml version = "1.0"?>
 <PID>
  <PID.3>
   <CX.1>A123456789</CX.1>
  </PID.3>
  <PID.5>
   <XPN.1>Smith</XPN.1>
   <XPN.2>John</XPN.2>
  </PID.5>
  <PID.7>19551212</PID.7>
  <PID.8>M</PID.8>
  <PID.11>
   <XAD.1>24 Some Street</XAD.1>
   <XAD.3>Dublin 25</XAD.3>
   <XAD.4>IRELAND</XAD.4>
  </PID.11>
  <PID.13>
   <CX.1>(01)1234567</CX.1>
  </PID.13>
 </PID>
```

Listing 2: HL7 version 3 message

HL7 version 3 was chosen as the messaging format for data exchange in this research as it is a well-defined standard that is used in hospitals worldwide. And, as will be discussed in Chapter 3, XML is used widely in the web-services community for exchanging data.

2.5 Clinical Analysis and Validation

Although this research is concerned primarily with test results produced in the home, a lot of research was carried out on the hospital and laboratory testing procedures as the system proposed interfaces with the laboratory test result validation system. Thus a description of the clinical laboratory testing process and result validation will be given. This will also help to highlight the difference between hospital testing and home-based testing and the problems associated with home-based testing. Section 2.2.2 has already given a brief introduction to the process of testing and validation in a laboratory and there now follows a more in depth discussion of the process.

2.5.1 Sample Acquisition and Analysis

For a doctor to manage a patient's illness, it is common practice to order specific tests on biological samples taken from the patient. These samples may be blood, urine, faeces, skin tissue, sweat, etc., and are taken from the patient by trained staff using good working practice. Figure 3 shows the tasks performed by a LIS when processing test requests. When a doctor identifies a clinical problem with a patient, s/he orders a set of tests on the patient. The results of the requested tests provide the doctor with useful information that will help confirm a diagnosis, monitor the effects of treatment and assess the patient's prognosis. A nurse or another clinician may take the biological sample from the patient, which is stored in the correct container for the sample type and clearly labelled with the patient's details and the details of the requested tests to be performed on the sample. The sample is transported to the laboratory and may be subjected to some pre-analytical treatments. Depending on the hospital, the sample information and requested tests may already be present on the LIS, otherwise this information is entered into the LIS and the sample is queued for analysis.

The clinical laboratory applies *gold standard* testing regimes on each sample to obtain a result or set of results. The gold standard is considered the most specific and sensitive test for this sample type and is completed strictly according to a workflow list. The analytical instrument usually controls the vital parameters in the functioning of the instrument as well as the test parameters and calibration data but they are all adjustable by the operator. The results are put into a temporary buffer, which can only be accessed by the operator [13]. The results are now validated either by the laboratory staff or a computer assisted validation system. If they are valid, they are made available to the doctor. This process is discussed in the following section.

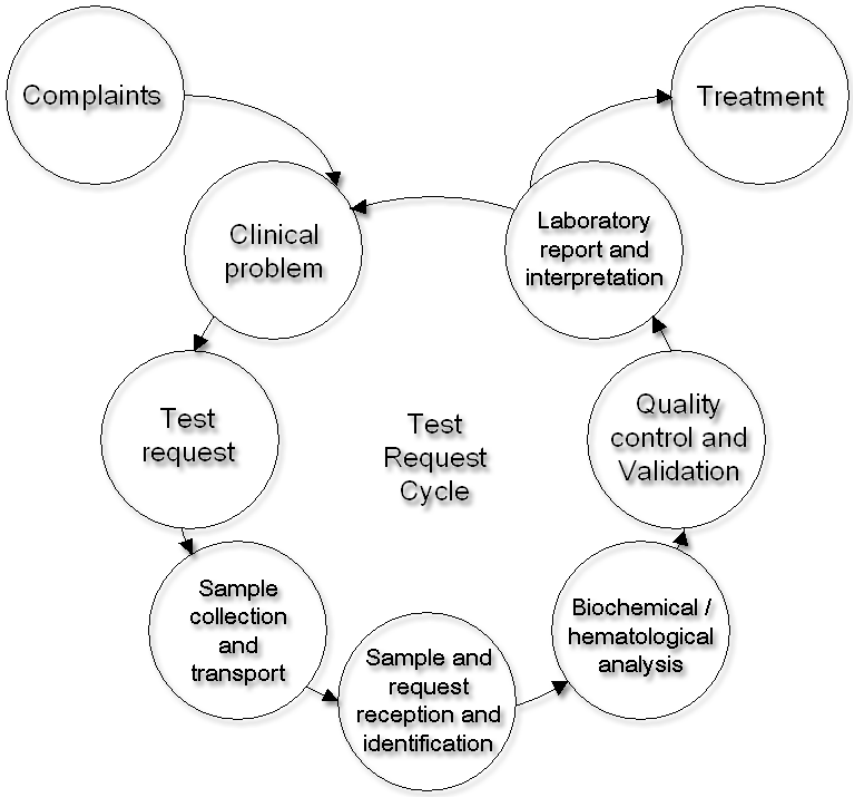

Figure 3: Overview of tasks performed by an LIS during a test request cycle [13]

2.5.2 Quality Control and Result Validation

In a clinical laboratory, work must be checked at every step in order to validate the results produced. The aim of validation is to prevent reporting of erroneous test results to clinicians. Manual result validation is very time consuming and can slow down the response of the laboratory. Computer assisted validation speeds up the validation process considerably by using either a rule based decision support system [21], historical data of test results [22] or both [23]. The best systems allow the laboratory staff to carry out a personal data check of all results that require verification [24]. Once a result is obtained the laboratory begins checking if this result is plausible for this patient in this instance. Result validation is generally divided into two areas, namely "Technical validation" and "Clinical validation".

Technical validation checks for instrument calibration, sample tolerance, accuracy verification, reportable range, reliability and certification. Within a batch of samples being tested there may be quality control (QC) samples which the analyser will

recognise as a QC sample and thus put the results from it in a separate file. The frequency at which an analyser is checked and calibrated with a QC sample is dependent on the equipment and the type of test it performs [25]. The results from the analysis of a QC sample allow the laboratory staff to identify any specific analytical problems. With this data the equipment can be calibrated ensuring the results are from a calibrated, certified process. Other technical checks are also carried out to verify that samples are not contaminated.

Clinical validation on the other hand is when the result is checked for plausibility. They ensure results are relevant to the patient and the complaint being investigated. Types of clinical validation include checking that the result value falls within age and sex related reference ranges, as there can be gender-dependent differences in test results. Delta checks compare patient's historical test results to the current result to detect analytical error and inconsistencies. The width of the historical time window in which to include previous results depends on the type of test. If the window is too wide the delta check loses sensitivity because the correlation between the actual and the historical test result weakens with a longer time interval [22]. Results exceeding critical limits during the validation process are reported immediately to the laboratory operator. An abnormal lab test result will normally require repeating to confirm the result. Prescribed drugs and diet can affect test results, thus causing them to fall outside the validation criteria. If this is the case, the laboratory operator can modify the validation algorithm to suit the individual case or can manually validate the result with the drug and diet information in mind.

Table 1 illustrates general QC tests undertaken by a clinical laboratory in the technical and clinical validation of a blood sample. When the checks are completed and if the sample results are still plausible the data is considered validated and populated into the electronic patient record. If at any stage the results are not plausible then the patient information is forwarded to the laboratory manager where a re-test of the sample can be authorized or a new sample requested.

Type of validation checks	Description
Basic validation checks	Data checked against age and sex related reference ranges, pathological limits as set down by the laboratory based on international standards.
Delta checks	The patient's previous results are compared with the current results using various techniques.
Analyser calibration	The instrument being used is checked to ensure it is calibrated and working correctly.
Instrument specific checks	Vary depending on instrument and analytical process used to generate the result. Often carried out either by the instrument itself or manually by the instrument operator.
Internal consistency checks	The consistency between pathologically and physiologically related variables is examined
Sample mix-ups	This results when one sample is given the identity of another, usually an adjacent sample on the workbench

Table 1: Validation Checks [23] (Modified)

2.6 Point of Care Testing

2.6.1 Introduction

Point of Care Testing (POCT) is defined as analytical testing performed outside of the clinical laboratory using a device or devices that can be easily transported to the vicinity of the patient [26]. It can be performed at the patient's bedside or close by in a centralised area within the clinical setting. POCT is growing rapidly and is being used in hospitals more and more because it reduces the turnaround time. Results are received more quickly than with a laboratory test and hence patients can be diagnosed and treated sooner. The diversity of testing locations for POCT is wide and includes hospital bedsides, ambulances, clinics, doctors' offices and patients' homes [27]. Although POCT is widespread in hospitals and is normally associated with a clinical setting, this research was concerned with patients who use POCT to manage their conditions at home and hence will concentrate on this aspect of it.

2.6.2 Patient Self-Testing and Home-Based POCT

Point of Care testing in the home is a rapidly growing area in the healthcare arena. It gives patients an opportunity to manage their own conditions and can reduce their

length of stay in hospital, minimising costs. There is also a cost reduction associated with the release of patients to their home for continuance of their healthcare. The benefits of patient self-testing include:

- No need for patient travel to the hospital
- Waiting time for results is reduced to the region of a few minutes
- Overall time to manage therapy is reduced
- Overall materials and personnel cost is reduced
- Patient freedom is increased
- Patient involvement and understanding of therapy is increased

With a rapidly ageing worldwide population, and the older share of the population set to double by 2030 [1], there is a need to increase patient self-testing at home in the health care system. There are a variety of ailments of the aged for which doctors can utilise clinical laboratory tests, and providing POCT to homebound patients could have a significant impact on their condition [28].The variety of test apparatus for patients who manage their conditions at home is as diverse as the ailments themselves, ranging from simple urine dipsticks and blood pressure arm cuffs to more complex ECG devices. Many diabetic patients also use blood-testing units, such as glucometers to check daily glucose levels. Blood coagulation (INR) meters are used by people taking anticoagulant medication such as warfarin. The management of diabetes in the USA costs $100 billion annually and has many secondary disorders associated with it. However, management of the condition with POCT would prevent many of these [28]. It is not only chronically ill patients and the ageing who use POCT, women during pregnancy and patients recently released from hospital may also have to monitor biological signs.

2.7 Point of Care and Home Testing Issues

POCT self-testing in the home raises issues that would not normally arise in a clinical setting. In hospital, tests taken by clinicians are validated by complex computerised validation systems before a diagnosis is made. Patients at home must often use the results of tests they take to determine medication dosage or monitor their condition, but these results do not undergo a validation procedure. Thus there is a need to implement a

system of result validation, either locally or by the hospital validation system itself, for people testing at home with POCT devices. In a 2001 study of hospital laboratories in America, lack of connectivity was the biggest issue for them and 72% had no connectivity between the laboratory information system and glucose meters outside the hospital [28]. The four major issues for patients who self-test at home with POCT equipment, which this research addresses are:

- The quality and reliability of the result are questionable as professional hospital staff do not oversee the sample acquisition and testing procedures. Testing quality achieved by patients operating blood glucose self-monitoring instruments is lesser when compared to a technician using the same instrument [29].
- The hospital may be unaware of possible complications in a patients' condition as they are not aware of the day-to-day results. If complications arise between visits, they may go unnoticed.
- There is no independent validation of the test results. Results from POC tests taken in the home are not clinically validated to ensure that they are plausible for that patient at the time of testing. Thus decisions made as a consequence of POCT may be erroneous, potentially leading to patient complications or death.
- Prescribed drugs and diet may interact and affect test results.

Hospital validation servers made aware of these issues can change validation parameters or the validation algorithm, thus tuning its accuracy and making comments tailored to the patients' current situation. This ensures a higher overall result quality. If point of care test results were validated in a similar fashion to hospital laboratory test results, as is proposed by this research, higher accuracy results and better patient care and advice could be achieved.

2.8 Point of Care Testing Equipment

An investigation into the different types of POCT devices available to patients for self-testing was carried out. The equipment investigated included simple urine dip-sticks, handheld blood analysers and more sophisticated self-monitoring stations. Many types of self-testing kits are available over the counter in pharmacies or indeed a supermarket. Details of the instruments investigated are available in the Appendix. Some of the more

popular POCT equipment and a description of the medical problems associated with them now follows.

2.8.1 Handheld Blood Analysers

2.8.1.1 Glucose Meters (Glucometers)

One of the most commonly monitored disorders using self-test equipment is diabetes. Diabetes affects nearly 200 million people worldwide and this figure is set to rise to 350 million by 2030 [30]. Glucose is a simple sugar that serves as the main source of energy for the body. The body's use of glucose hinges on the availability of insulin, a hormone produced by the pancreas. There are two main types of diabetes, Type 1 and Type 2. People with Type 1 diabetes are insulin dependant and are normally diagnosed by the age of 30. Type 2 diabetics account for about 90% of all diabetics. This form of diabetes normally occurs later in life and is generally due to factors such as lack of exercise, being overweight, a family history or high blood pressure [31]. Poor control of diabetes can lead to serious complications such as blindness, stroke and renal diseases [28]. In addition to dietary controls, ongoing diabetic treatment revolves around daily glucose monitoring and control. Type 1 diabetics must self-check their glucose levels and inject themselves with insulin several times a day. Type 2 diabetics usually self-check their glucose once or twice per day and control their diabetes with diet and exercise. They may also take oral medications to stimulate insulin production in the pancreas [31].

Glucose monitoring is performed using a glucose meter or glucometer as it is popularly known. A glucometer is normally a small handheld meter with a digital display. There are many makes and manufacturers of these devices [32], [33], [34]. Nearly all glucometers allow the patient to store past results with time and date and some allow them to store results on their computer. The method in which the glucometer interfaces with the computer is device specific but in [12], Cronin outlines a standard method for interfacing a POCT device with a PDA. To perform a glucose reading, a small drop of blood, 0.3µl – 7.0µl obtained by pricking the skin with a lancet, is placed on a test strip which is inserted into the meter. The meter analyses the blood drop on the strip and the glucose value in mmol/l (mg/dl in the US) is displayed within a matter of seconds. The normal blood glucose range for a patient who has been fasting is 5.6 to 6.9 mmol/l [31].

2.8.1.2 Prothrombin Time (INR) meters

Anticoagulants are medications used to prevent clotting in blood and blood vessels. Warfarin is an anticoagulant medication that is administered orally. Warfarin is given to patients who have a tendency to form blood clots (thrombosis), patients with cardiac problems or patients who have artificial heart valves. Warfarin has a very narrow therapeutic range meaning levels in the blood that are effective are very close to levels that can be problematic (Figure 4). Dosing of warfarin is further complicated by the fact that it is known to interact with many other medications and other chemicals which may be present in appreciable quantities in food (including caffeine). These interactions range from enhancing warfarin's anticoagulation effect to reducing the effect of warfarin [35]. An overdose of warfarin is potentially very dangerous and can lead to severe bleeding and even cerebral hemorrhage. Thus, regular blood monitoring of patients receiving warfarin is imperative for the safe and effective use of it [36]. This is normally carried out in a warfarin clinic in the hospital and the patient has to travel to the hospital at least once per week for testing and review of their dosage. However, self-testing for anticoagulation is on the increase and there are more accurate POCT devices available today [37], [38]. Studies have shown that patients who self test for anticoagulant medication can achieve outcomes as good as or better than at a warfarin clinic [9].

Figure 4: Warfarin risks [37]

The gold standard method for monitoring warfarin therapy is prothrombin time, expressed as an international normalised ratio (INR), collected via venipuncture [36].

Prothrombin time is the time it takes plasma to clot after addition of 'tissue factor' (obtained from animals) [39]. Tissue factor is a substance present in tissues necessary for the coagulation of blood. INR was devised by the World Health Organisation to standardise the results of prothrombin time as differences can occur between different batches and manufacturers of tissue factor. INR is the ratio of a patient's prothrombin time to a normal (control) sample and is calculated by the formula:

$$INR = \left(\frac{PT_{test}}{PT_{normal}} \right)^{ISI}$$

The ISI is the International Sensitivity Index and indicates how the particular batch of tissue factor compares to an internationally standardized sample. The reference range for prothrombin time is 12 - 15 seconds and for INR is 0.8 - 1.2. It should be noted at this point that a mobile phone application, such as the ones to be discussed in Chapters 4 and 5, would be more than capable of performing a calculation such as this one if necessary.

2.9 Summary and Conclusion

This chapter provided a review of the current standards and practices in hospitals with regard to data communication systems, laboratory sample testing and test result validation. Patients who travel to hospital to have samples analysed in the clinical laboratory can be assured that the results they receive are the most accurate for the specific test. This is not the case for patients who monitor their condition at home as certified clinical staff do not oversee the testing process and the results produced are not subjected to any post-analytical validation. This post-analytical validation is part of any laboratory test process and is very important for ensuring that the result is high quality and plausible for the patient, given their condition.

However, as seen in section 2.6.2, there is no doubt that there are many advantages of patient self-testing. And, with the increase in the technology available for patient-hospital communications, POCT equipment and telemedicine, more and more patients will be managing their illness' at home in the near future. This chapter highlighted the need to provide a link between these patients and the hospital that is responsible for their treatment and safety. This will ensure that the hospital will have a record of the patients' medical condition and that the results will be subjected to a post-analytical validation process.

3 Technology Overview

Before discussing the design and implementation of the system proposed in this thesis, it is important to give an overview of the technical tools researched and used in designing the system. There are a number of different software tools and technologies used in the system including a database management system, client-server programming tools and communications device considerations.

The system of remote validation was built using client-server architecture with a database on the server side. This chapter will discuss the tools that were used to build the system and aims to provide the user with an overview to the relevant technologies. After a brief introduction to mobile phones and their computational possibilities, the Java platform and Java 2 Micro Edition will be discussed. Next, the server side tools will be described including Java 2 Enterprise Edition, Java Servlets, web services and the MySQL Relational Database Management System.

3.1 Mobile Phone Capabilities

Applying the available systems and services of the communications industry into the healthcare arena could greatly benefit patient care. This is why mobile phones were chosen as the most practical device to use for the applications to be described in this thesis.

3.1.1 Mobile Handsets

Mobile phones are everywhere; the worldwide number of mobile subscribers passed 1.5 billion in 2004 and is set to reach the 2 billion mark by mid-2006 [40]. Most mobile phones available today are more than the simple voice-centric devices that were originally developed. They are now complex data communication terminals. Many are equipped with additional wireless technologies such as IrDA, Wi-Fi and Bluetooth. Memory sizes are increasing and are normally in the tens to hundreds of megabytes and processor speeds are typically a few hundred MHz [41]. In fact many are more powerful than the desktop computers of the 1990's and are continually evolving to an "always

on" form of network computer, capable of availing of Internet services continuously. These advancing features mean mobile phones can be used to provide a user with a rich client application, capable of performing computation and data storage locally and of interacting with remote servers to avail of additional services. Aside from the technological advances of mobile handsets and their supporting networks, their abundance is rapidly growing. Mobile handsets have become one of the most popular computer based consumer devices in the world today (see Table 2) [42]. Although a modern computing device, mobile phones are still relatively easy to use and the familiarity that people have with them means learning to use an application on the phone is a simple task.

3.1.2 Mobile Networks

The Groupe Spéciale Mobile (GSM) was formed in the early 1980's in order to develop the specification for a mobile communications network capable of supporting the many millions of subscribers likely to turn to mobile communications in the years ahead. This flexible and reliable mobile communications network would facilitate the increasingly international nature of business [43]. In 1992, after its official launch, it was renamed the Global System for Mobile communications. The GSM network is a cellular network, consisting of many base stations, each covering a geographical area. These cells partition the available frequency range and reduce the range of each base station in order to reuse the scarce frequencies as often as possible [44]. In Europe the system uses frequencies in the 900MHz and 1800MHz bands and the channel bandwidth is 200kHz. An innovative feature of GSM is roaming, which provides subscribers with the ability to make calls and send/receive data whilst travelling outside the coverage of their home network. This means that a patient who might be using a condition management application, such as those described in this thesis, could travel abroad and not worry about their contact with the hospital.

GSM only allows data connection rates of up to 14.4 kbps. Whilst this is adequate for voice calls, it is rather low for data transfer. In 2000 General Packet Radio Service (GPRS) was introduced to accommodate the need for increased data transfer rates in the GSM network. Sometimes referred to as 2.5G, GPRS was developed to bridge the gap between the 2G and future 3G systems. Ideally data rates of up to 171.2 kbps are

achievable but more realistically single user throughput is likely to be 112 kbps [45]. With GSM, a channel is dedicated to a user for the duration of a call. But for data applications using GPRS, the available bandwidth is maximised by sharing channels and switching packets of data to the required destination [46].

While 3G handsets can combine the functionality of a phone with that of a PDA, they also have broadband connections to the Internet and are able to achieve much faster connection speeds than GSM [47]. The standard in Europe for the development of 3G is known as Universal Mobile Telecommunications System (UMTS), which can support both voice and data with bit rates up to 2 Mbps [46]. With data rates like this, 3G has the potential to revolutionise the mobile computing industry. As well as large amounts of raw data, both video and audio transfer is possible with 3G and this could be of enormous benefit to patients based in the home and to the general field of telemedicine.

Device	2003 sales in millions
Mobile phones	> 500
Desktops and servers	128
Laptops	36
Portable compressed music players	24
PDAs	10.4
Tablet PCs	0.6

Table 2: Computing device sales for 2003 [42]

3.2 Mobile Phone Technologies Considered

Some mobile technologies were investigated in order to find a suitable method of transmitting results reliably to the hospital. The system discussed in this thesis requires that the communication of patient information should happen in a secure and timely manner. The Java 2 Micro Edition platform was chosen as it was the best solution for providing a secure and generic application capable of running on any Java enabled mobile phone. It will be discussed in the next section in more detail, the rest of this section will discuss the other technologies that were considered as possible solutions.

3.2.1 Short Messaging Service

The Short Messaging Service was first considered as a possible means of data transfer between the patient's device and the hospital server. SMS was originally introduced to

GSM networks to notify users when they had voicemail messages. Now SMS is used widely for simple applications such as downloading ringtones and information services. SMS messages are generally thought of as text messages but can be a byte stream of data up to 160 bytes in length. However SMS messages can be concatenated to allow more data transfer as part of the same transaction. Messages sent to and from mobile phones and computers are maintained by an SMS Centre (SMSC) [48]. The SMSC stores the message and then forwards it when it can be delivered. However there is no guarantee that the message will be delivered instantly and as the system being proposed by this research is of a time-critical nature, SMS is not a viable solution. In addition to this, SMS does not allow for any computation on the phone and is being superseded by Wireless Application Protocol (WAP) and 3G technologies.

3.2.2 Wireless Application Protocol

The Wireless Application Protocol (WAP) provides a standardised method for linking wireless devices such as mobile phones to Internet services [49]. It is an open standard that lets wireless devices easily interact with services and allows users to access the Internet [50]. The latest version of WAP, WAP 2.0 adopts existing Internet standards. WAP incorporates a relatively simple micro-browser into the mobile phone. Content is developed in XHTML, which is similar to standard HTML meaning people familiar with fixed internet standards can easily develop content for wireless devices. The content can be stored locally or be served by XHTML compatible web sites. It uses the GPRS network, which allows for the highest possible data transfer rates in the standard 2G GSM network. So, for example, it could be used by patients to access a hospital server, submit test results and have relevant information returned to them, both textually and graphically. The server could be responsible for all of the computation and storage of data. Although this is a better solution than using SMS it is still not a satisfactory solution for the requirements of this research. XHTML is purely a mark-up language and does not provide direct access to the hardware platform. Thus, no computation or off-line storage of test results would be possible.

3.3 Java Technologies

Both the client and server applications designed and developed in this thesis were written using Sun Microsystems' Java 2 Platform. The Java platform is based on the

power of networks and the idea that the same software should run on many different kinds of computers, consumer gadgets, and other devices [51]. A key feature of Java is that it is platform independent, meaning a Java program can run on any type of computer that has a *Java Runtime Environment* (JRE). JRE's are available for almost every type of computer including Windows based computers, Macintosh computers, Unix and Linux machines, mainframes and mobile phones. There are three editions of the Java programming language, each one designed to cover a different area of business logic. This allows for the Java platform to be used for developing everything from simple smart card and thin client applications to complex multi-tiered enterprise applications. The relevant editions can be chosen by developers to suit the needs of the application. The three editions of the Java 2 Platform are the Java 2 Platform Standard Edition (J2SE), Enterprise Edition (J2EE) and Micro Edition (J2ME). Figure 5 shows the different editions of Java and an example of the type of device that it may run on. J2SE, the most common version of the Java platform, contains the essential libraries for developing client-side and general-purpose applications. J2EE is the superset of J2SE and is the Java platform that is targeted at enterprises to enable development, deployment, and management of multi-tier server-centric applications [52].

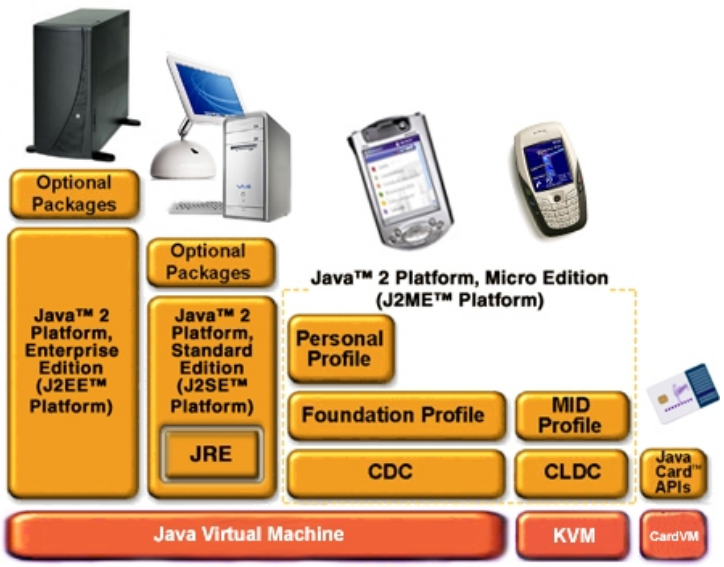

Figure 5: Java 2 Platform [53](modified)

J2ME is a highly optimised edition of the Java 2 Platform and is a subset of J2SE. It is targeted at small, standalone or connectable devices such as mobile phones to enable development and deployment of applications on them.

Within each edition of the Java 2 platform, there are different Java Virtual Machine (JVM) implementations, optimised for the type of systems at which they are targeted. The K Virtual Machine (KVM) is a JVM optimised for resource constrained devices, such as mobile phones [54]. As outlined in [54], the following characteristics are shared among the three Java editions.

- **Portability:** As Java byte-code is interpreted by the JVM, applications written in Java will run on similar types of systems. They adhere to the Java mantra, Write Once, Run Anywhere.
- **Security:** This is a major issue in today's web-centric environment. Before any application is executed by the JVM, a byte-code pre-verifier tests its code integrity. Systems are also protected by the Java sandbox security model, in which applications execute. The sandbox is a protected environment that prevents access to protected resources. Java also supports standard encryption techniques.
- **GUI:** All editions of Java have APIs for developing rich user interfaces. This proved important for this research project as the client applications that were designed for mobile phones were to be patient focussed and easy to use.
- **Networking:** Java programs are 'network agnostic' and can exchange data with servers over any network protocol.

For this project, the mobile phone client programs were written using J2ME. These programs contained the logic for taking the patient's result data and transmitting it to the server. They also stored results locally so they could be viewed off-line. These applications communicated with the health provider server in two ways. Either via Java Servlets or Java web services, both of which are a part of the J2EE specification. The server side programs provide access to other services and applications and with the database to store and validate the patient's results.

3.4 Java 2 Micro Edition

3.4.1 Introduction

Java 2 Micro Edition (J2ME) is a set of technologies and specifications developed for small devices with limited memory and computation possibilities such as mobile phones. J2ME is a subset of the Java 2 Standard Edition (J2SE) and hence makes it easy for people who are familiar with Java technologies to develop applications for wireless devices. J2ME shares many of the same characteristics of its sister editions, including, platform independence, and network awareness. It also has a rich set of libraries for developing GUIs. As discussed in the last section, Java is a portable, platform independent language. Developing the applications with J2ME ensures that they would run on any mobile device that supports Java, and there are literally hundreds of these from all of the major manufacturers [55]. The J2ME architecture contains the Connected Device Configuration (CDC), the Connected Limited Device Configuration (CLDC) and the Mobile Information Device Profile (MIDP) standards. The high level architecture of a typical J2ME device is shown in Figure 6.

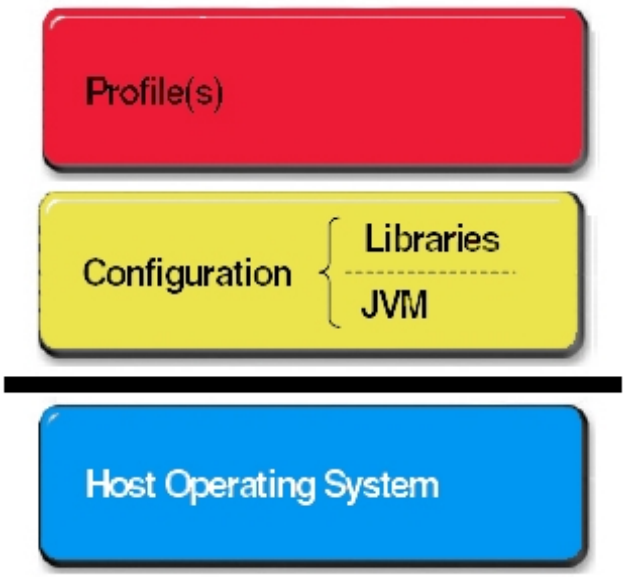

Figure 6: Software layers of a J2ME device [56]

The CDC is aimed at high-end consumer devices with a large range of user input capabilities, memory budgets in the range of 2 to 16 Mbytes, 32 bit processors and persistent network connections such as TCP/IP [57]. The CLDC and MIDP were

specifically designed for low-end consumer devices such as mobile phones and are the minimum specifications that all J2ME enabled mobile devices are expected to support [53]. As these are the type of devices targeted by this research the CLDC an MIDP will be discussed in greater detail. Figure 7 shows how the various components of a J2ME device fit together. These individual components will be discussed further in the following sections.

Figure 7: Overview of J2ME components [58] (modified)

3.4.2 Connected Limited Device Configuration (CLDC)

According to Riggs et al [53], 'a J2ME Configuration defines a minimum platform for a horizontal category or grouping of devices, each with similar requirements on total memory budget and processing power. The CLDC is a set of APIs that allow developers to design applications for devices with limited resources such as small screens, and limited memory and processing power. It assumes that the virtual machine, the libraries and the applications will fit within a 160 – 512 kilobyte memory budget. The J2ME CLDC contains the basic packages for common operations, which are a subset of the J2SE packages [59]. The current version of the configuration is the CLDC 1.1 (JSR-139), which has been made more J2SE compliant generally, including added support for

floating point numbers, improved Calendar and Date classes and improved error handling capabilities. Due to these additions the minimum memory budget was raised from 160 to 192 kilobytes [56].

3.4.3 The K Virtual Machine (KVM)

At the heart of all Java enabled systems is the Java Virtual Machine, which executes the Java byte code, this is what makes Java platform independent. J2ME CLDC devices have a more compact Virtual Machine, tailored for the memory restrictions of the device called the K Virtual Machine (KVM), where the K stands for 'kilo' as its memory budget is measured in tens of kilobytes [53]. CLDC and MIDP applications run on top of the KVM. The KVM is designed to be as compliant with the standard JVM as possible. However, in order to accommodate the strict memory restrictions of mobile devices, it has many features eliminated and a more limited set of error handling classes. In case of a runtime error that is not supported by the CLDC Error classes, it will throw the nearest CLDC-supported superclass. The KVM also supports low-level security to ensure an application cannot harm the device it is running on and will reject invalid class files via its class file verifier [56].

3.4.4 Mobile Information Device Profile (MIDP)

A profile is defined as a set of API's that supplement a configuration and provide capabilities for a specific market type or device [60]. The MIDP supports devices that implement the J2ME CLDC. The MIDP APIs allow developers to deal with mobile device specific issues including user interfaces, local storage and making HTTP connections. The current version of the profile is MIDP 2.0 (JSR-118) and the specification [61] is available from Sun Microsystems. It outlines all of the features and requirements of the MIDP and the following information is based on this document. Some of the minimum hardware and software characteristics specified for MIDP compliant devices include:

- At least one user input mechanism such as a one-handed keypad or touch screen.

- At least 256 kilobytes of non-volatile memory for a MIDP implementation, 8 kilobytes of non-volatile memory for application specific persistent data and 128 kilobytes of volatile memory for Java runtime.
- A display size of at least 96x54 pixels.
- Two-way wireless networking possibly intermittent with limited bandwidth.
- Simple sound and tone capabilities.
- A minimal kernel to manage the device's hardware.
- Ability to read and write to non-volatile memory to support the Record management system.
- Read and write access to the device's wireless networking protocols to support the Networking APIs

The MIDP APIs also define the application lifecycle and how an application is controlled. A MIDP application is called a MIDlet and is one that uses the APIs defined by the CLDC and the MIDP, this type of application will run on any J2ME enabled device.

3.4.5 Additional API's and Packages

In addition to the MIDP profile, which is included in all J2ME implementations, vendors can write and create their own device specific packages. These may be for accessing features on the phone such as a built in camera, or the vendors' menu structure. Additional packages provide the developer with more options and greater functionality, allowing them to build better applications. Sun also provides optional J2ME packages for various features including Bluetooth connectivity (JSR 82) and Mobile 3D graphics (JSR 184). Java Specification Request 172 (JSR 172), the J2ME Web Services API (WSA), extends the J2ME platform to support web services. However, before talking about this in more detail, web services and server side logic will need to be discussed further.

3.5 XML

XML which stands for eXtensible Markup Language is a structured system for organising documents. An XML file is a tagged data file. The tags in the document

define the structures and boundaries of the embedded data elements. The structure of an XML document is similar to that of a HTML web document but XML enables developers to customise tags. One of the primary purposes of XML is to facilitate the sharing of data across different systems, particularly systems connected networks. In general, an XML document will contain *elements*, *attributes* and *entities*.

- An element is something that describes a piece of data and is comprised of markup tags and the elements content for example:

  <patient_name>Joe Public</patient_name>

- Attributes are used within an element to provide additional information about the element and are contained within the starting tag of the element. So, for example, type is an attribute that describes the result type of a patients test result:

  <test_result type="blood glucose">

- An entity is a virtual storage of a piece of data that you can reference within the XML document. So department is an entity in the following example:

  <!ENTITY department "Clinical Chemistry">

When an application receives an XML document it must extract the element and attribute data and perform further processing if necessary. Reading and extracting the data from an XML file is referred to parsing the data and is performed by an XML parser.

XML is very important to the work in this thesis from two points of view. Firstly, as this work employs client-server architecture, a structured well-defined method for passing information needs to be used. XML is used extensively on the web and with web services. Section 3.6.3 describes further how XML is used with web services and Java. Secondly, as outlined in Chapter 2, Health Level 7 (HL7) messages for hospital communication are written using XML. As this research requires a client to connect with a hospital based server, the documents must be structured in a manner that is recognisable to the hospital information system. Thus all result and patient information

transferred between the hospital and the patient is embedded in XML documents to ensure the data can be interpreted reliably at both ends.

3.6 Server Side Tools

The technologies discussed so far in this chapter have all been concerned with client side computing. However, connecting patients with a hospital server is one of the primary goals of this research and in order to develop a full proof of concept system, a server would have to be implemented. A server is responsible for handling requests from client applications and returning the relevant response information. It can communicate with databases for storage and retrieval of data and can collaborate with programs on other computers and servers anywhere in the world. This section describes the middle-tier and server architectures researched.

3.6.1 Common Object Request Broker Architecture (CORBA) [62]

CORBA is a middleware platform whose specification is maintained by the Object Management Group (OMG). In short, CORBA enables communications between distributed objects. The key feature of CORBA is that the client object does not need to know where the server object is located, what language it is implemented in or what platform it is running on. A client only needs to know the logical structure of an object it wishes to use as detailed in its interface. It invokes the object across a network to experience its behaviour but it has no knowledge of the implementation of the object. The OMG Interface Definition Language (OMG IDL) is used to define the types of objects available to clients by specifying their interfaces. An interface provides a framework for objects and consists of a set of the operations and their parameters. IDL is the means by which object implementations tell clients what operations are available and how to invoke them. Each programming language, including Java, has a mapping to allow a program to access CORBA objects. However, although CORBA has many desirable features and is well supported by J2SE and J2EE, it did not prove to be a good solution for this research. The main reason being that CORBA depends on a TCP/IP connection to provide interoperability between objects, via its Internet Inter-ORB Protocol (IIOP) and there is no requirement for support of Mobile IP [63]. As this

research requires communication over wireless networks, CORBA was deemed unsuitable for communication between the client and server.

3.6.2 Java Servlets

In order to access the service of result validation and storage to the patient, there must be a web-based gateway that the client application can contact to upload and process the test results. As mentioned earlier, J2ME applications are capable of making network connections, and in particular HTTP (HyperText Transfer Protocol) and Secure HTTP connections, to servers. In order to create a fully working implementation of the system, a web-tier server had to be developed to handle the client requests and return the right response.

Java servlets are special Java applications that run on a web server, often referred to as the servlet container. The web container used for the implementation of this work was the Apache Tomcat web server, discussed later. Java servlets are part of the web-tier of the Java 2 Enterprise Edition (J2EE) and provide client applications with powerful logic and the ability to interact with databases. Clients connect to the server using standard protocols, which are available on most client platforms such as the HTTP GET and POST methods. Thus the server logic and applications need only be written once and be placed on the server for access by many clients, allowing developers to create efficient thin-clients [59]. Servlets are supported through the Servlet API, an extension to the Java programming language. Within this API the **javax.servlet.Servlet** interface is defined, which must be implemented when creating a servlet. Servlets using the HTTP protocol implement the **javax.servlet.HttpServlet**. This is the servlet type used in the system designed in this thesis as J2ME supports HTTP connectivity. This will be described in more detail in Chapter 4.

As the client applications developed for this research are to run on mobile phones, they need to be so-called ‘thin-clients’ and the processing they need to do is kept to a minimum meaning that solid and reliable server programs had to be developed.

3.6.3 Java Web Services

Web services are an emerging area of distributed computing. A web service is a web accessible application that exposes a public interface useable by other applications over the web [59]. This makes it possible for one application to access another application on a computer anywhere in the world, regardless of operating system or architecture. More relevant to this work, it means that a thin client running on a mobile phone can find a service on the network and use it as though it is a local service. Web services make use of the following standard protocols [64]:

- **Simple Object Access Protocol (SOAP):** An XML-based protocol that provides an envelope for exchanging object data on the Internet. It provides a fully extensible mode of communication between software systems.
- **Web Services Description Language (WSDL):** Another XML-based protocol, WSDL is used to describe a remote service and facilitate application-to-application communication. A WSDL file describes what a service does, how to invoke its operations, where it can be found on the network and its interface.
- **Remote Procedure Call (RPC):** An Internet protocol that allows data to be exchanged between systems. With SOAP, it lets one application call methods on another application without needing to know anything about the underlying network.

Web services are platform and language independent and they ensure seamless interaction between requesting client and responding web service. Once a web service is written and ready to be deployed, the developer must define a service description and describe the applications methods in the form of a WSDL file. The URL of this WSDL file is made available to client developers. The client developers use the WSDL file with a stub generator program, which provides them with the code necessary to make remote calls to the service. The client developer incorporates the generated stub code into their application and can make calls to the web services methods. This stub code handles all of the network and RPC connections so, to the client, it seems as if the service is local, or within the application itself.

3.6.4 J2ME Web Services APIs (JSR 172)

Section 3.4.5 described how J2ME had some additional profiles and APIs. One such set of APIs is the J2ME Web Services APIs (WSA), which enable J2ME devices to be web service clients. These were developed within the Java community process as Java Specification Request 172 (JSR 172). These APIs give wireless clients two optional packages: one for remote service invocation (Java API for XML-based RPC or JAX-RPC) and one for XML parsing (Java API for XML Processing or JAXP) [64].

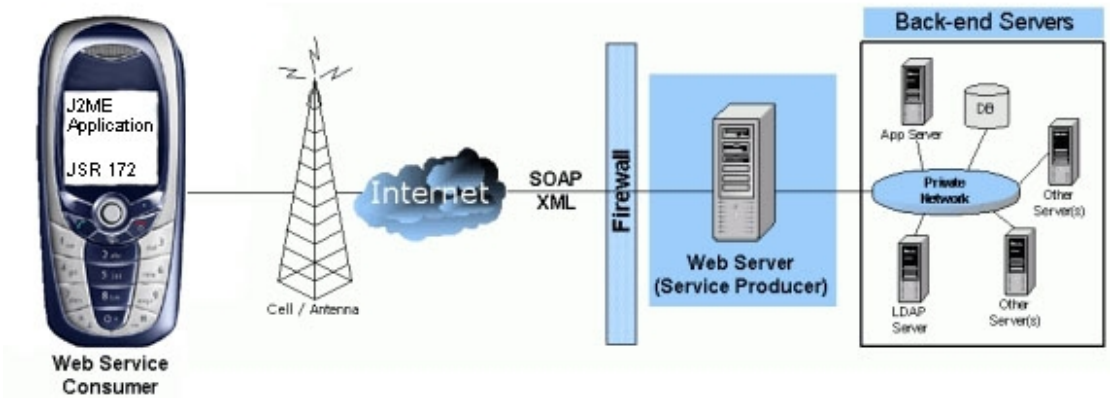

Figure 8: J2ME web services architecture [65] (modified)

The web service architecture has three elements, which are shown in Figure 8. First, a network aware application that runs on a wireless J2ME WSA enabled device. The application includes a JSR 172 stub to communicate with the network. Second, the wireless network and communication protocols including HTTP and SOAP. Finally, the web server containing the web service that the client wishes to invoke, this may also provide access to back-end servers and applications.

3.6.5 Apache Tomcat Web Server

As mentioned in section 3.6.2, Java Servlets are applications that run on special web servers called Servlet containers. Tomcat is one such Servlet container that has proven to be reliable in both test and production environments [66]. It serves as Sun's official reference implementation and is fully compliant with the Servlet specifications published by Sun. The Tomcat server is open-source and free to distribute. All Java Servlets and web services discussed in this thesis were hosted on either the Apache

Tomcat server version 5.5.9 or the Apache Tomcat server version 5.0 for integration with Java Web Services Developer Pack 1.6.

3.6.6 Relational Database Management System

A relational database management system (RDBMS) is a system where data is stored in a collection of tables related to each other through common data values. Tables of independent data can be linked to one another if they contain columns of data that represent the same data value [67]. Databases use SQL (Structured Query Language) as the tool to create, manage and query data. MySQL is a popular open source RDMS and was chosen for the implementations in this thesis for the following reasons:

- MySQL is a medium-scale RDBMS, more than capable of handling the task at hand.
- It is a very fast performing system with a slimmer feature set to aid performance, yet has most of the features present in larger scale systems.
- MySQL is open source, so it is free and hence there are no issues with licensing.
- Commercial systems, such as Oracle, require far more system resources to run, i.e. memory, CPU and disk space. This could mean that a separate machine would have to be set aside to maintain the system.

3.7 Summary and Conclusion

An overview of mobile technologies and networks was given in this chapter. Mobile phones have become a popular and familiar consumer device to over a billion people worldwide. Consumers are now using their mobile phones for more than just voice calls and the potential for application development in this area is great. The advances being made in the ICT arena means mobile phones are becoming powerful, (broadband) network aware computing devices, giving great potential to application developers. Coupling this with the issues discussed in Chapter 2, providing patients who self-manage their condition with remote assistance, means a mobile application can provide a practical and reliable solution. The technology most certainly already exists for such a solution, it is now a matter of applying it to the health arena and exploring the

possibilities. The next chapter introduces some possible solutions using the technologies discussed in this chapter.

A number of software technologies were also introduced in this chapter, which were applied in the design and implementation of the proof of concept system. The Java 2 platform and its three editions were introduced and discussed as they feature heavily in these designs. J2ME provides a powerful and rich set of features for mobile phone application development and the enterprise edition of Java allows powerful server programs to be written. Other important areas discussed included the formatting of data with XML, the MySQL DBMS for handling data on the server web servers, namely the Apache Tomcat server.

Now that mobile phone technologies and technical tools relevant to this research have been introduced and discussed, the reader should have a good background for understanding the design and implementations of the proof of concept systems to be discussed in the following chapters.

4 System Design

The main aim of this research project is to provide a reliable method for patients to transmit test results to the hospital so they can be validated and stored in their patient record and to return advice or instructions based on the result values. So far the background to patient testing, the issues associated with it and technologies available have been discussed. It has been established that a solution utilising Java technologies will provide the best system that will work across a wide range of mobile devices. This chapter discusses the proposed system, which provides a possible solution for patient test result validation and storage as stated above. The design of three different solutions and the different technologies they employ, as well as the issues addressed and considerations made will also be discussed.

4.1 Overall Requirements

4.1.1 Overview

Three different versions of the system were designed and these will be discussed individually in later sections. Before examining specific sections of the design, a discussion on the overall requirements and goals of the system will be given.

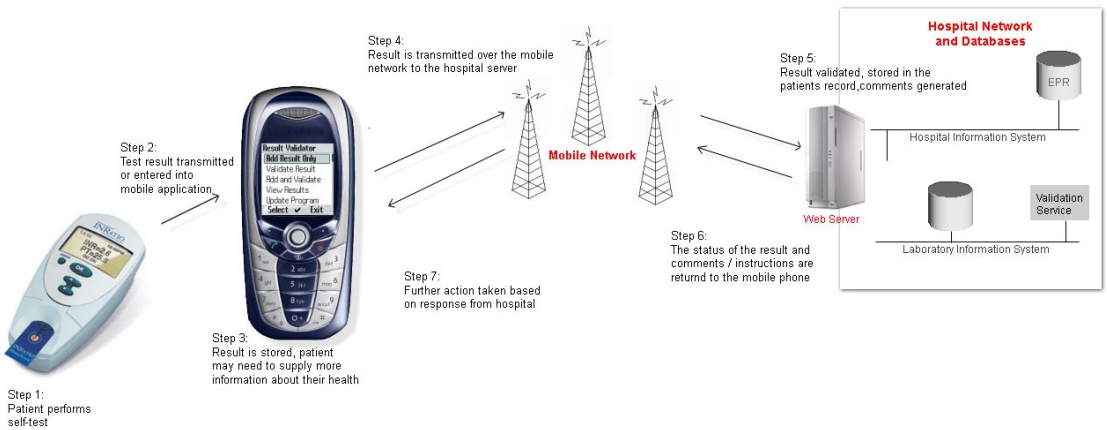

Figure 9: General overview of required system

Figure 9 depicts a generic overview of what the proposed system aims to cover. The patient, using a POCT analyser, performs a test in the home on a specimen taken from

the body. The analyser produces a result, which may be displayed on a screen. Next, the analyser transmits the result data to a mobile phone. The transmission of result data from an analyser to a mobile device using Bluetooth was explicitly researched by Cronin [12] and has been omitted from this research for clarity. Once the result data has been received by the mobile phone application, it is stored on the phone. If necessary, the application can collect additional information from the user that is relevant to their condition, by asking simple questions. Further processing and simple validation can be performed on the phone before sending it to the hospital. At this point the information is prepared for transmission to the server and is wrapped into XML format. The application connects to the hospital's web server, sends result information and waits for a response. The web server now performs the necessary tasks to get the result validated and stored in the Electronic Patient Record (EPR). The server generates the response to be returned to the patient, which may include advice, instructions, medication dosage or a request for the test to be retaken. This information is sent back to the user application, which then displays the message to the user and updates its own record of the result.

4.2 Overview of Alternative Designs

Three different designs of the system are introduced here. Each version of the system aims to provide the patient with a service similar to that outlined in the previous section (Figure 9). However, each one is different in some way or another to the other two.

Version	Validation Process	Server Communication
1	Phone	Java Servlet
2	Server	Java Servlet
3	Server	Remote procedure call (web service)

Table 3: Three versions of the remote validation system

Table 3 shows the differences between the three versions by highlighting the location of the validation process and the method in which the client application communicates with the hospital server. This will become clearer as each version is described individually in the coming sections. A full discussion of all aspects of the design of each version will be given in individual sections later.

4.2.1 Version One

For certain test result types, the method for validation of the result is quite simple and requires only certain information, such as past test result values, whether the patient has been fasting, current medication and the sex, age and weight of the patient. This kind of information can easily be obtained by an application via a simple questionnaire given to the patient or from local storage of a patient profile. As stated in Chapter 2, modern mobile phones are powerful mini-computers and capable of quite complex computation. If a validation algorithm is simple enough to only require information such as that stated above, then it is conceivable that an application running on a mobile phone could contain such an algorithm.

Allowing the result to be validated on the phone gives the distinct advantage that the patient does not have to wait nearly as long for the outcome of a result validation as it does not have to make a network connection or rely on the validation service in the hospital being available at any given time. It also helps to reduce the workload of the hospital system. In a hospital, results to be validated are entered into a work list and processed chronologically. A greater number of results waiting in such a work list, means a longer wait to validate the result. This, coupled with the fact that wireless networks are relatively slow in comparison to today's fixed networks that people are used to, means that the turnaround time from a patient sending a result to receiving the hospital response could be relatively long.

Thus, the first system design contains a validation algorithm on the phone in order to provide a faster system. When a user enters the result into the application, it is validated on the phone. If validation of the test taken requires more information, they may be asked some questions. For example, a patient with a glucose monitor may be asked to supply information about diet. The application will return the outcome of the validation to the user along with some advice or dosage information. However, another goal of this research is to provide the hospital with the test results so they are kept informed of the patients condition and have a record of them when the patient returns for a check up. So, once the patient has received the outcome of the validation, the application connects to the hospital's web server via a Java servlet and sends the result data. The application

running on the server stores the patient's result and informs the user (and application) that it has been received so the connection can be closed.

4.2.2 Version Two

The previous version is a good solution for validation algorithms not requiring complicated computation or many external resources. However, for more complex algorithms and rule bases, a more powerful computer will need to be used. Thus the result data must be transmitted to the hospital for validation as well as storage. Validating the result on the hospital server ensures the result is validated by the gold standard system that applies to all hospital results. It also can allow for a laboratory professional to view the details of the result if necessary and provide better and more dynamic information to be returned to the patient.

The second version of the proposed system allows the result to be sent to the hospital via a Java servlet. The servlets are accessed over a HTTP connection that the client application makes, and the result data are transmitted to them. Once a result is received in the hospital, the servlet parses the result data and stores it in a database. Running separately on the hospital network is the validation service. This is always checking the incoming results in the database and when it finds one that is not validated it performs the relevant validation on it. Once these are validated they are stored in the patient record and the updated information along with the generated instructions is sent back to the client application.

4.2.3 Version Three

This version of the system is a distributed computing solution. Its only similarity to the second version is that the actual validation of the result is performed on the hospital server. The key difference is the method it uses to communicate result data back and forward to the hospital. The previous two versions communicate with the hospital via a servlet running on the web server. For this method, the server is running a web service, which has methods made public for validating and storing a result. Using a remote procedure call (RPC), the client calls the method on the server to validate the result. The actual connection to the remote service is handled by the generated stub code so to the

user it feels as though it is calling a local method. The main purpose of designing this version was to investigate an alternative method for connecting the client with the services offered by the hospital.

4.3 Result Data Format

As mentioned in Chapter 2, the information and data to be sent from the patient's device to the hospital and back, was based on Health Level Seven (HL7) messaging standards. HL7 version 3 uses XML format messages to structure clinical data. HL7 messages are made up of various segments, each one describing a specific data type. The structure of the result data from the analyser, to be sent to the hospital, was based on the OBX segment. This segment is the observation / result segment. An OBX segment of a HL7 message can contain many attributes including the reference ranges and the date of the last similar result. However, as this information is to be sent over a wireless network, a simplified version was designed, to keep the message length to a minimum but still containing key attributes. The format of the result data was kept the same for all three versions of the system. The result data was represented in the programs by a class called **ResultData**.

Attribute Name	Data Type	Description
Result ID	Long integer	A unique identifier created using timestamp
User ID	String	An unique ID to identify the patient
Result Value	Float	The numerical value of the result
Units	String	The units of the result
Test Code	String	A code defining the type of test
Status	Integer	Whether it is validated, valid or invalid
Date Produced	Long integer	The timestamp of produced date
Date Validated	Long integer	The timestamp of validated date
Notes	String	Notes, Comments, instructions for patient

Table 4: Attributes of result data

The attributes are extracted from this class and wrapped into an XML document for transmission to hospital. The web server can then add the additional attributes and convert it to an actual HL7 result segment before passing it on to be processed further in the Hospital Information System. The hospital server returns an XML message of the same type but has updated the attributes relating to the validation. Table 4 shows the

attributes that were chosen to represent a result for transmission. The attributes chosen were ones that are relevant to the POCT analyser and provide enough information for the hospital to be able to perform a search in the database to retrieve any other necessary information. The User ID attribute is used to identify the patient to avoid sending the patient's actual patient ID over the wireless network.

4.4 Web Server Database

Result data received by the web server application needs to be stored in a database. This allows it to be accessed by other services so it can be validated and stored in the patient's record. The database can be used for matching the patient's user details that they submit, to the actual patient details from their EPR. Also, as the client application only transmits certain key attributes of the result data, the rest of the data relevant to the patient and the test result can be obtained using information from this and other databases so it can be prepared for use in the hospital information system. A very simple database with only a few tables is sufficient for this purpose.

The entity-relationship (ER) diagram shown in Figure 10 shows the tables that are needed for a database on the server. The *incomingdata* table is to store the result data that is being sent from the client applications. It contains attributes for result data similar to those described in the last section, the PatientID attribute can be retrieved from the *patient* table. Users of the system will have to have their account details stored and this can be done in the *users* table, which also contains a reference to the patient's details through the PatientId foreign key. Finally, the ranges table is used to store information about test result reference ranges for the first version of the system, so the client can check if it has the most recent data for validation.

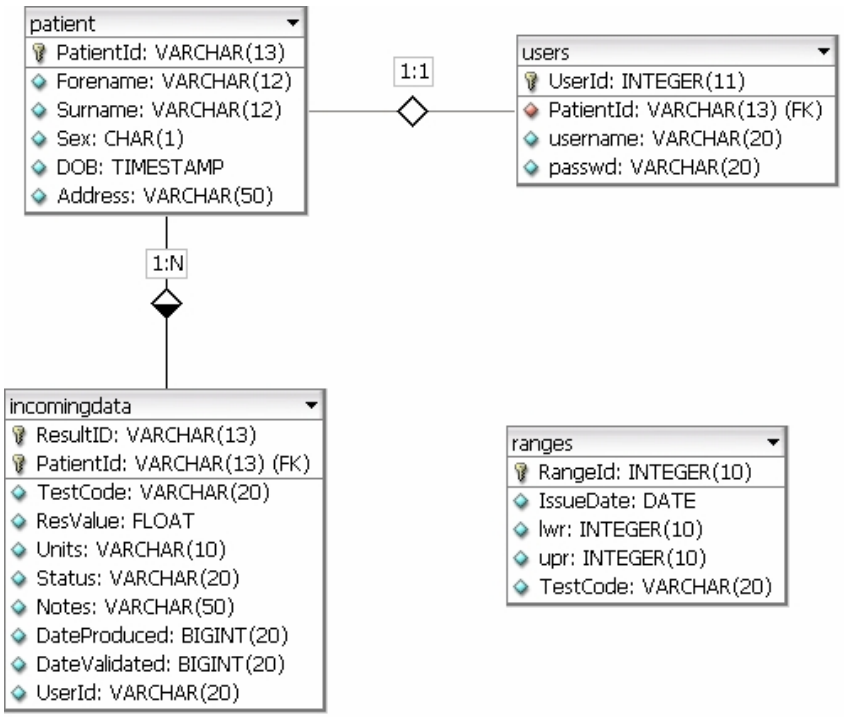

Figure 10: E-R Diagram for web server database

This simple database could be made far more complex for a real implementation, but is adequate for the proof of concept implementations to be discussed in this thesis.

4.5 Design of Version One

As previously mentioned, this version is designed to validate the result on the phone. The main components in this version are the J2ME client MIDlet, the web server running the Java servlet that the client communicates with and a database to store the incoming data. This database acts as a temporary store for the result data received, the data are then moved into the hospital information system and on to the patient's EPR. An overview of the system is shown in Figure 11. On the left hand side of the figure is the client application, which transmits the result to Servlet application where it is processed and sent to the EPR in the hospital system on the right.

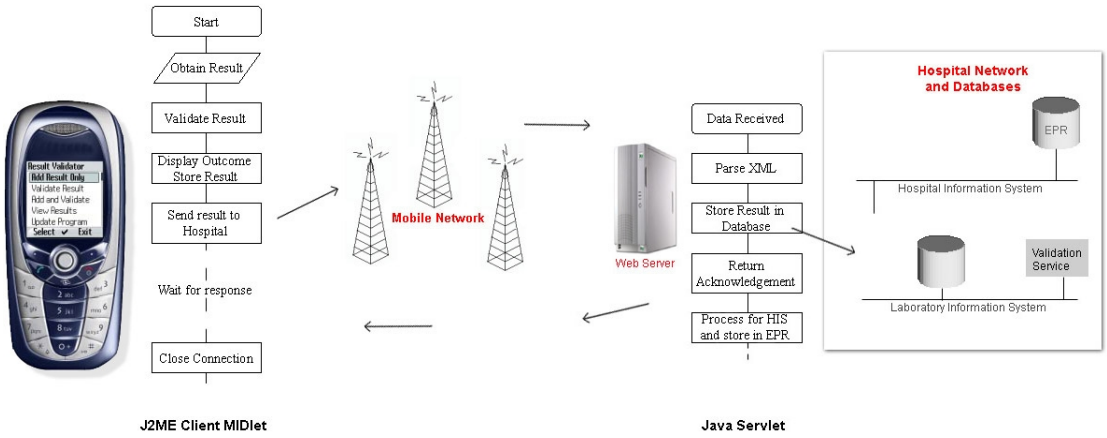

Figure 11: Overview of version one

4.5.1 Client MIDlet

This is the client application for the patient to manage their test results, it is capable of storing result information locally in a simple RMS data store. The main menu provides the patient with options to add a new result, view stored results, delete results and to check for updates for the program. When a result is added, it is validated on the phone and stored locally. When the user selects the view results screen they are returned a list of the current results stored in the phone. The details of each can be viewed and they can transmit the result to the hospital server if this has not already been done.

The Class Diagram is shown in Figure 12. Table 5 gives a description of each of the classes used by the client application. **Version1MIDlet** is the main MIDlet class which controls the user interface and user input and output. The result data can be represented in the **ResultData** class. When a new result is entered, the time is noted to the nearest millisecond, this is a 13 digit long integer and is also used when creating a unique result ID. Once a result is stored and validated, it has to be to uploaded to the hospital's web server. All network methods for doing this are accessible via the **ServerConnect** class. The MIDlet communicates with the server using HTTP POST. The HTTP POST request method means a client can send data of unlimited length to the web server as well as retrieve information from it [53], which is necessary in this application.

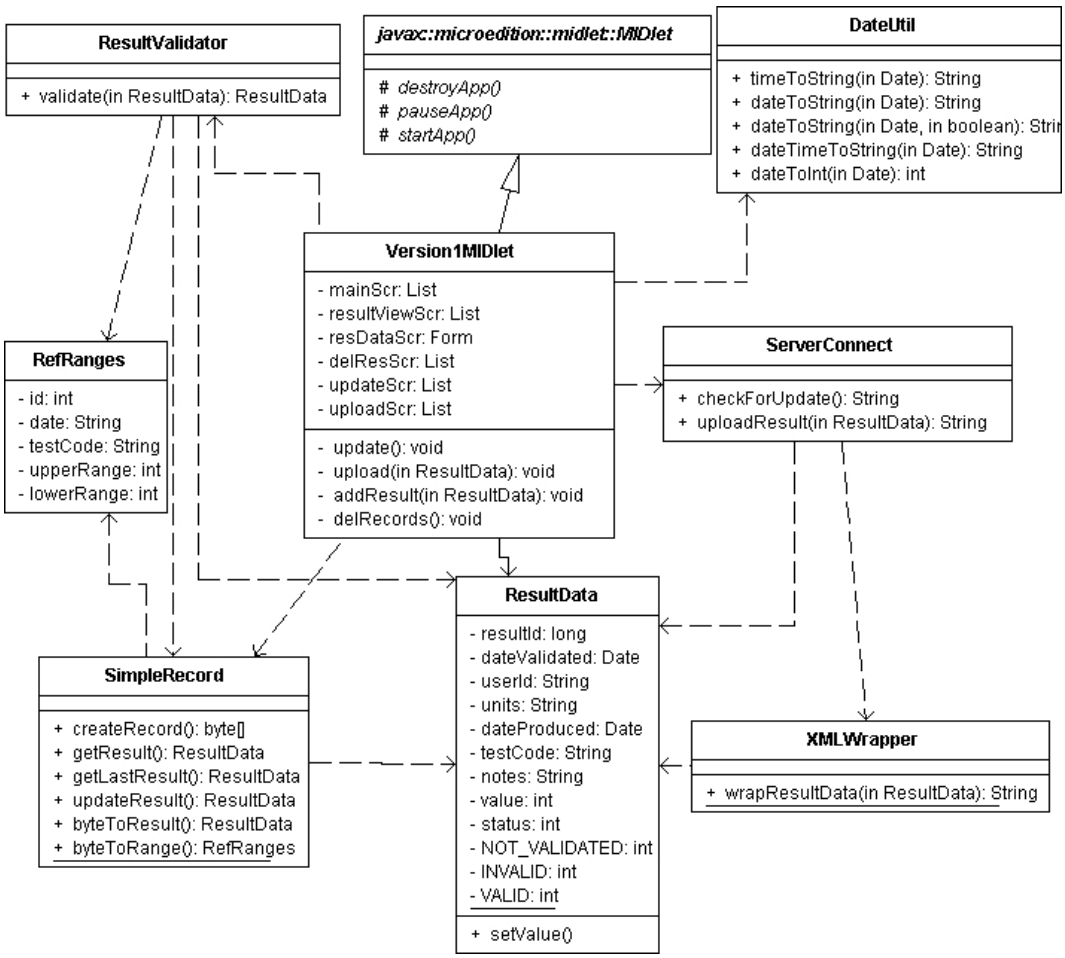

Figure 12: Version 1 MIDlet Class diagram

Class Name	Description
Version1MIDlet	This is the main MIDlet for controlling the application. It contains the methods for displaying menus and information to the user, accessing results, uploading information to the server.
ResultData	This class is a simple representation of a result as discussed in section 4.3.
ResultValidator	This class contains the methods to validate a result. All of the rules etc. that are needed to validate a result will be accessed here. The **validate()** method is called by the MIDlet class to validate a result.
SimpleRecord	For controlling the RMS store, it defines the size of a record, and contains methods for creating and retrieving records from the phones memory.
ServerConnect	All of the network methods needed to connect to the server, such as uploading results, are in this class.
XMLWrapper	A class with methods to convert result information into an XML document string so it can be transmitted to the hospital web server.
DateUtil	Contains useful methods for formatting dates.

Table 5: Version 1 MIDlet Classes

An RMS Record Store is used in this program to store the result data. The SimpleRecord class contains the functions to add, remove and update results. Each result record occupies 41 bytes of data. The first 13 are used to store the result ID. As this number is also the time in milliseconds, it can be used to retrieve the date of the result using the java **Date** class. The rest of the bytes in the record are used for the result value, the notes and for a status integer (valid, invalid or not validated), which is also used to tell whether the result has been uploaded to the server or not. Data such as the normal reference ranges for a test result are also stored in another RMS store. This information is used by the **ResultValidator.validate()** method when validating a result. The MIDlet is designed with an option to connect to the server and download new rules and ranges for the validation method, so ranges could be specifically tailored to an individual patient.

4.5.2 Server

The web server in the hospital is responsible for receiving the patient result data and transferring it into the hospital information system. In this version of the system the client communicates with the hospital via a Java servlet. The java servlet class diagram is shown in Figure 13. The servlet that the client invokes is the **ResultServlet**, which implements the **javax.servlet.HttpServlet** method. As mentioned in the last section, the client invokes the servlet using the HTTP POST method and so this servlet uses the **doPost()** method. The byte stream of result data from the client is captured in a **ResultData** object using the **parseResult()** method of the **XMLParser** class. This can then be put into the incoming data table in the database using the **DatabaseConnector.insertResult()** method. When this has completed the servlet returns a status integer, which can be interpreted by the client to indicate either the result was received OK or an error has been encountered.

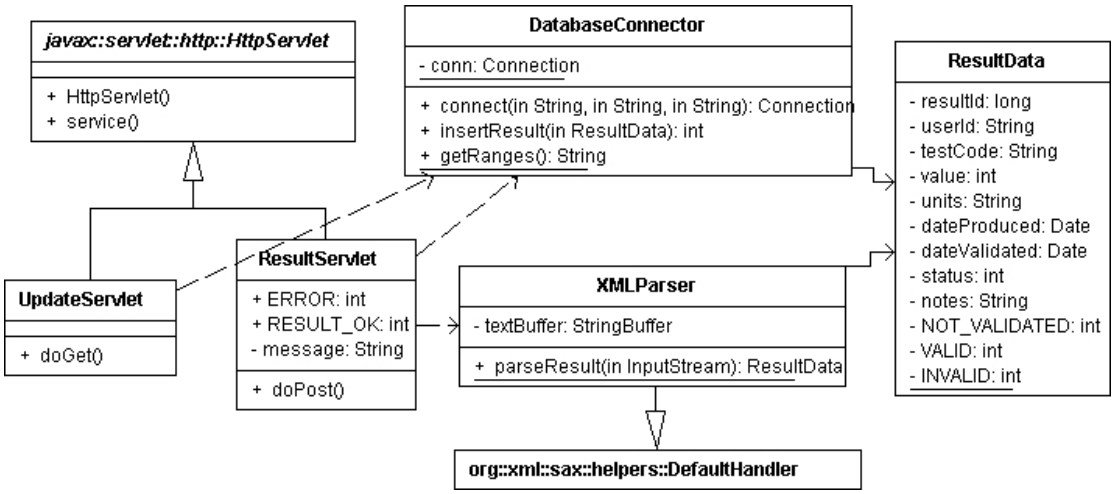

Figure 13: Class diagram for the web server

The result in the database can now be extracted by another service running on the hospital server, further processed if necessary and then entered into the electronic patient record. The reference ranges that the client application uses can be updated and are retrieved when the user invokes **UpdateServlet**. This queries the database to see if there are newer ranges or if special ranges for the individual patient are available, and returns them if found.

Class Name	Description
ResultServlet	This is the servlet that the client invokes for uploading the result to the hospital.
UpdateServlet	The servlet invoked to retrieve updated reference ranges if available.
ResultData	A class to hold the result data.
DatabaseConnector	This class contains all methods for accessing and entering data in the database.
XMLParser	This class contains the method to parse the XML data sent from the client.

Table 6: Version 1 servlet Classes

4.6 Design of Version Two

This version transmits the patients result to the hospital web server for validation. An overview of the system is shown in Figure 14. It is similar to the previous version in its

structure, with a client MIDlet that communicates with the server by invoking Java servlets.

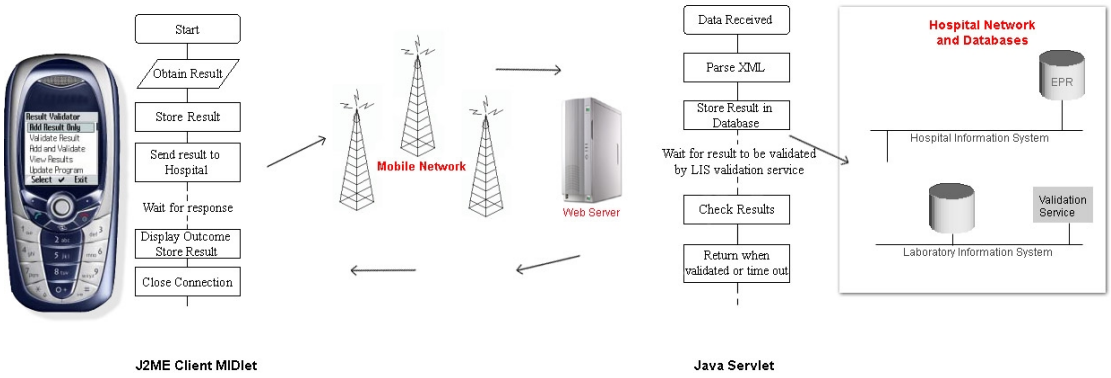

Figure 14: Overview of version two

When the result is validated, the servlet returns the updated result data to the user along with the instructions. The client updates its record of the result in the phones memory so it can be viewed off-line.

4.6.1 Client Application

The client application class diagram is shown in Figure 15. Many of the options available to the user will be to upload and validate the result, view stored results and delete records. When a result is entered, it is stored, wrapped to an XML format document and then transmitted to the hospital server. As the result is validated remotely, there are no validation classes and hence no need to have the option for updating validation rules. The remote validation means the user of the client application will have a longer wait for response from the server, so some additional classes have been included to show the network activity.

An overview of the MIDlet classes is shown in Table 7. This application contains the **ResultParser** class for extracting the result information sent back from the server, which includes the updated notes and instruction message for the patient. The **ServerWait** class is used to inform the patient that the result is being uploaded and takes control of the interface away from them to avoid the result being transmitted multiple times by an impatient user.

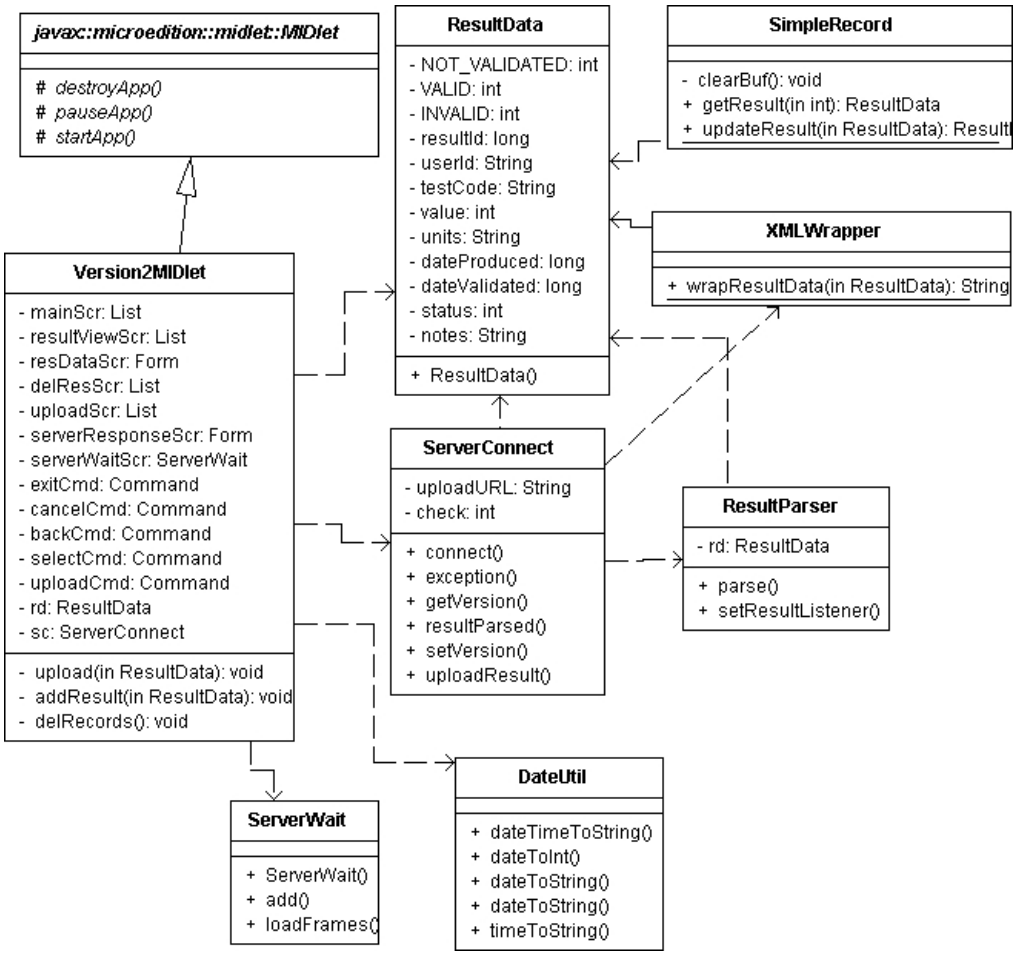

Figure 15: Class diagram for version 2 client

Class Name	Description
Version2MIDlet	This is the main MIDlet for controlling the application. It contains the methods for displaying menus and information to the user, accessing results, uploading information to the server.
ResultData	A class to for accessing and storing result data as discussed in section 4.3.
ServerWait	This class is for displaying network activity to the user when uploading the result.
SimpleRecord	For controlling the RMS store, it defines the size of a record, and contains methods for creating and retrieving records from the phones memory.
ServerConnect	All of the network methods needed to connect to the server, such as uploading results, are in this class.
XMLWrapper	A class with methods to convert result information into an XML document string so it can be transmitted to the hospital web server.
ResultParser	The result data returned from the hospital server will be wrapped as an XML document. This class contains a method for extracting the information to a **ResultData** object.
DateUtil	Contains useful methods for formatting dates.

Table 7: Version 2 MIDlet Classes

4.6.2 Server

The server in version 2 is also accessed using Java servlets. There is only one main servlet in this version as there is no update facility. When the result data is received and parsed it is stored in a database. The program then waits for the result to be validated and continually checks the status of the result in the database. In the mean time, a separate service in the hospital sees that a new result has arrived for validation and validates it, updating the patients record and the database that the servlet is waiting on. When the servlet sees that the result has been updated it retrieves the new information and wraps the data in an XML document. This information is then returned for interpretation to the client MIDlet.

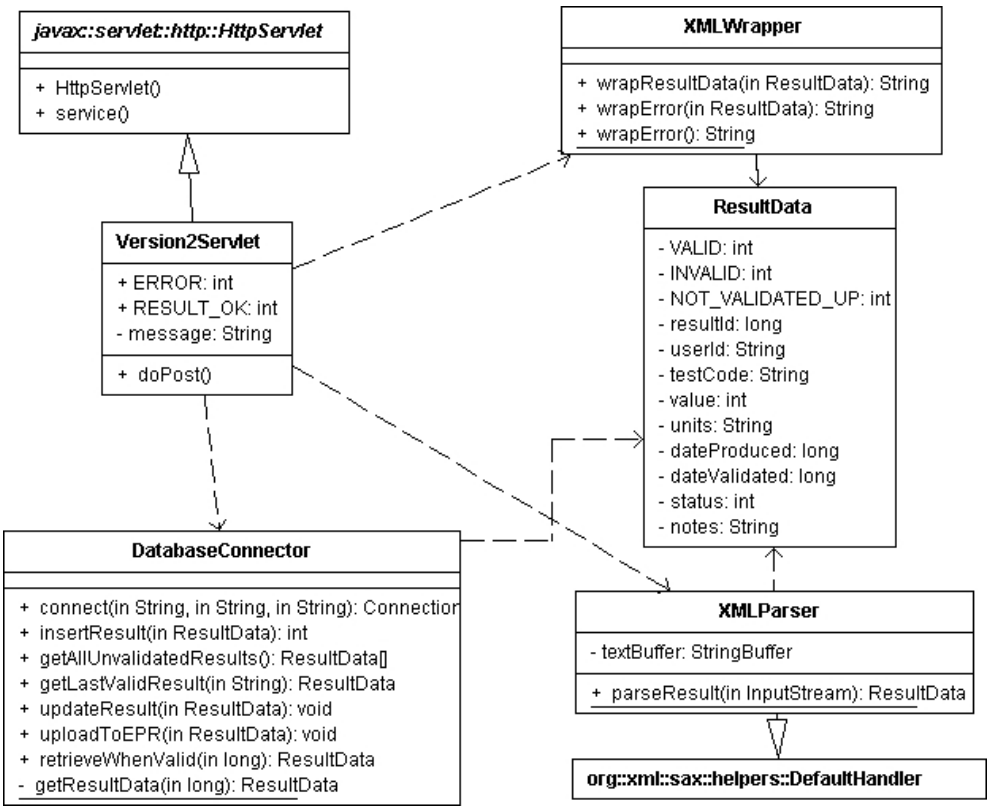

Figure 16: Class diagram for version 2 server application

If the result is not validated and updated in a timely manner, the server sends a time-out error message back to the client. This error is also wrapped as an XML document. In this case, the user will be asked to resend the data and the server will check to see if it has been validated yet. Table 8 gives an overview of the classes shown in Figure 16.

Class Name	Description
Version2Servlet	The servlet that the client invokes for uploading the result to the hospital.
ResultData	Class to access and store result data.
DatabaseConnector	Contains all methods for database access.
XMLParser	Used to parse the XML result data sent from the client.
XMLWraper	Similar to the client class of the same name. Result and error data extracted from the database for transmission is wrapped as XML.

Table 8: Overview of version 2 Servlet classes

4.7 Design of Version Three

This version of the system adopts a different method for client/server communication, namely Java web services. As discussed in Chapter 3, the Java web services APIs (JSR-172) provide J2ME application programmers with packages for remote service invocation (JAX-RPC) and XML parsing (JAXP). Version 3 is designed so that the hospital web server provides a web service that a remote client can invoke using remote procedure calling (RPC). The service is of course the point of care test result validation and storage and the provision of advice and notes based on the result values. As with the previous two versions, the design of the system was broken into two parts, the web service and the J2ME client MIDlet. The basic steps to creating a web service are: Write the web service interface class, write the implementation of this class, create and publish the Web Services Description Language (WSDL) file which is used by client developers for gaining access the service and finally, deploy the service on the web server. The client application is developed using the WSDL file and the service is invoked using RPC.

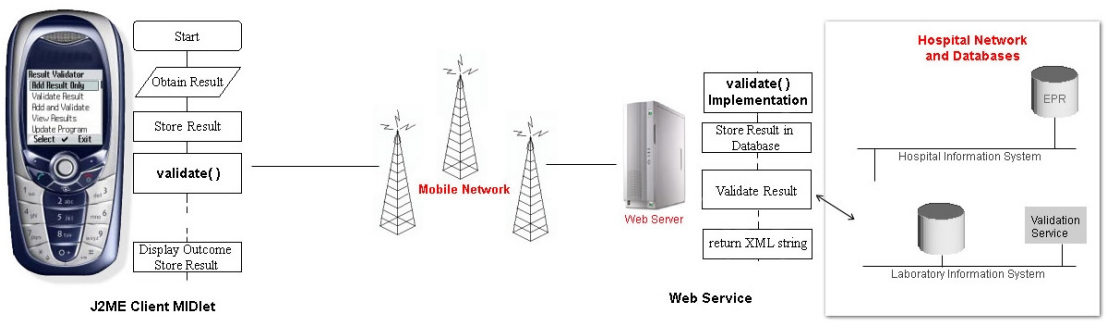

Figure 17: Overview of version 3 system

4.7.1 Web Service

The interface class defines the methods on the web service that clients can invoke. For this implementation of the system, the validation service is declared through the **Validate** interface class. The client calls the service through the **validate()** method, passing it the User ID, result ID, result value, test code and a Boolean indicating whether the user has fasted or not. The web service returns a string, this is actually an XML document containing the result data.

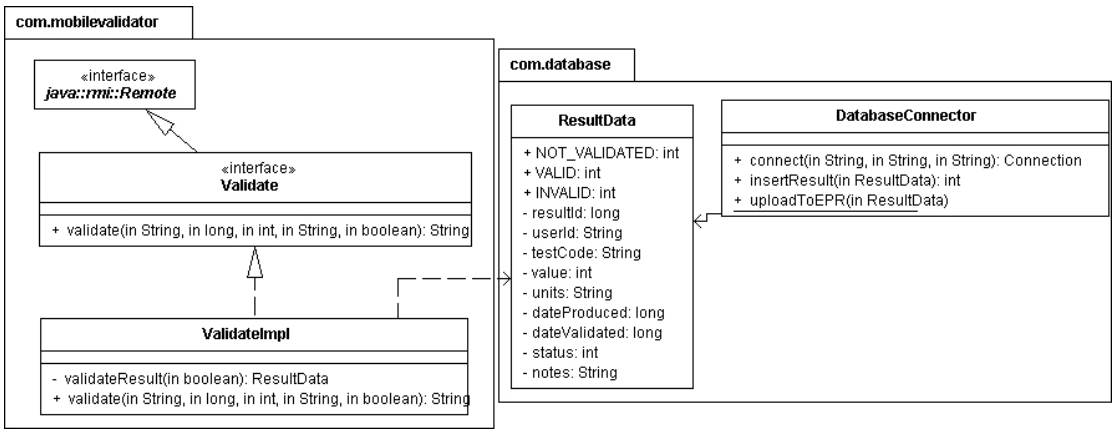

Figure 18: Web service interface and implementation

The **Validate** web service is implemented with the **ValidateImpl** class, these are shown in Figure 18. For this design, the database related classes including the **ResultData** class are in a separate package. The implementation of the **validate()** method creates a result data object using the information passed to it and by extracting additional data from the database. The result is then validated and stored in the database exactly as is done in the previous two designs. Next the XML result data document is constructed and returned to the client calling the service in the form of a string. This version uses a different method to wrap the result data to XML, namely, the Document Object Model (DOM), which is a Java API for working with XML documents. When the application is compiled as a web service, stubs, ties and a WSDL file are generated. Stubs and ties are low-level classes that the server uses to communicate with the client. Finally the web service is deployed to the web server and is ready for clients to access it.

4.7.2 Client Application

The J2ME client application provides the same functions as the previous versions, so a user can transmit the result for storage, remote validation and also view past results and advice from the local memory of the phone. The main difference in its structure is how it communicates with the server. Using a stub generator tool in the J2ME wireless toolkit, with the WSDL file of the web service as the input, the client side stub-code is generated. This stub is a local java object, which handles communication between client and server. The MIDlet makes local method calls to this stub and then the stub calls the web service on the server.

The class diagram for the MIDlet is shown in Figure 19. The four classes in the **com.mobilevalidator.validationservice** are the stubs that are automatically generated by using the WSDL file. These classes are implementations of interfaces that are available in the JAX-RPC package of the J2ME web services APIs. All of the networking and data transfer logic required to contact the web server is encapsulated in this package, and to the user of the system it seems as though a local service has been called. To use the web service, the method **validate()** in the **Validate_PortType_Stub** class is called. This has the same input and return types as the method on the web server as described in the last section.

The application represents result data in the same way as the previous designs with the **ResultData** class. The MIDlet parses the XML result received from the server using the **parse()** function which is part of the JAXP package in the web services APIs. **ResultParserHandler** contains the **getResultData()** function which returns a **ResultData** object using the parsed information so the result information can be easily accessed for further processing and storage. The **EntryForm** class is for obtaining user input.

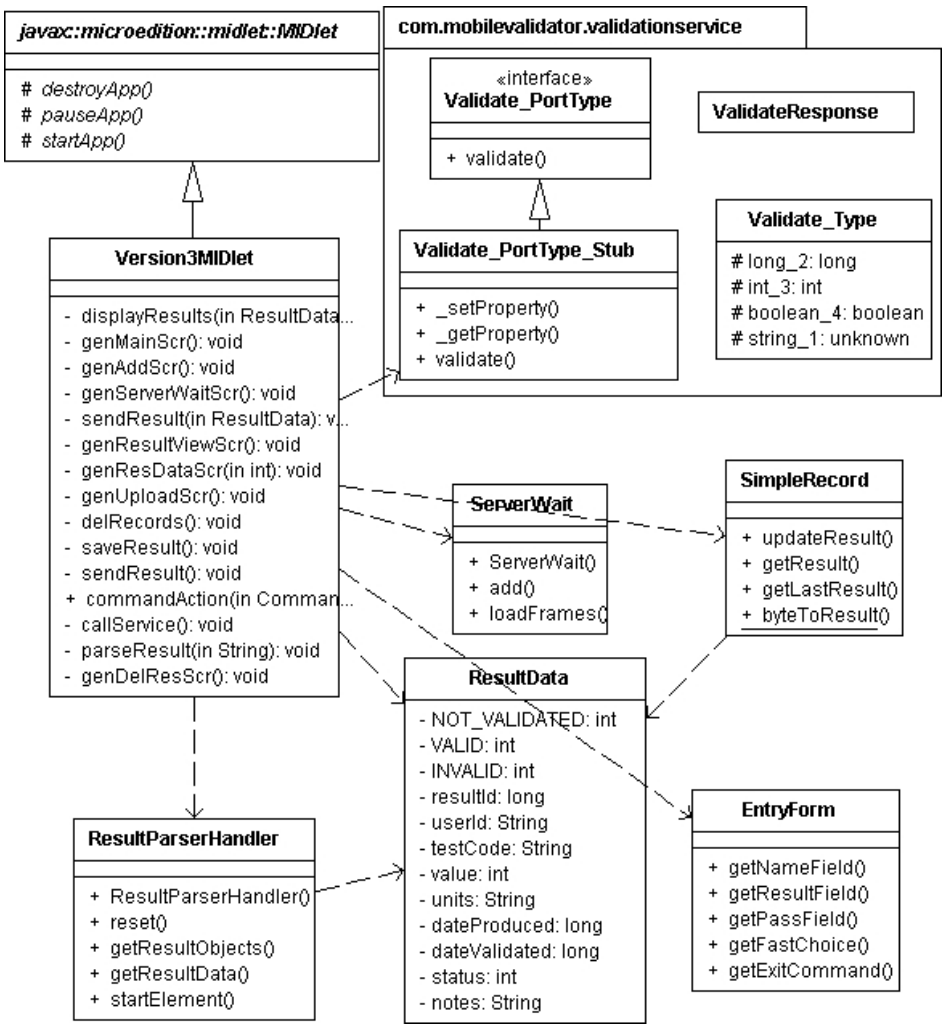

Figure 19: Class diagram for version 3 client application

4.8 Summary and Conclusion

Three alternative versions and their design have been introduced in this chapter. Each version uses a different method to provide a patient with a service to validate and store the results of their tests in their EPR. The first version provides the patient with a system to validate the test result on the phone and then connect to the hospital server to upload this result. This means the user will not be subjected to delays that may occur on the back end server or with the hospital validation system if there is a big backlog of results for validation. Allowing the result to be validated in the hospital system however, as in the second version, ensures the result is validated to the same precision as any test taken within the hospital would. It also allows for much more complex validation algorithms and it can access information from a far greater data pool, such as

the laboratory database and any data from the EPR. The third version also gives the patient access to the hospital server to validate results, but communication with the web server is managed through a web service. The client makes a remote procedure call straight to a validation method on the server and is then returned the outcome of the validation.

The decision of which version provides the best solution for the system depends on factors such as how complex the validation algorithm is, turnaround speed, i.e. time taken from entering data to receiving the response from the server, and which one creates the least network traffic. To investigate these factors an implementation of each version was carried out to provide a proof of concept for each version. The three implementations were then tested with these factors in mind and results for each version compared. These implementations are discussed in detail in the next chapter.

5 Implementation and Test

5.1 Introduction

With the design of the system established, a proof of concept implementation was developed. This involved three separate implementations, one for each version of the system that was discussed in the last chapter. The three implementations were tested and compared to one another and the advantages and disadvantages of each will be discussed in this chapter. The three implementations are based on a patient who has diabetes and enters measurements from a glucose meter. However, a system for any type of condition could be developed based on those described in this chapter.

In describing the implementations of the client MIDlets, there will often be references made to the **commandAction()** method, mainly when discussing user input and the current user display. All of the client MIDlets described in this thesis implement the **javax.microedition.lcdui.CommandListener** interface which defines the **commandAction()** method. This method is implemented so that the MIDlet can respond to command events from the user, and is checking (listening) for user input all the time.

Each version of the system was built on the previous one, and in addition to the main version modification being made, improvements and modifications were added to each subsequent version to help reach a better overall system. For example, version two implements a better user interface for display when there is network activity. This is also inherited by the third version but it adds a better entry form for the patient, which requires them to provide user identification. All of the modifications made along the way can be applied to the previous versions, but in order to highlight the development of the systems in this discussion, they were left as originally implemented.

5.2 Server Database

The database designed for the web server was implemented first, using the MySQL relational database management system (RDMS). The main purpose of this database is to store incoming result data so it can be accessed by other services for further processing. It is also designed to store information about users of the remote validation system with a link to their patient details so any other relevant patient data can be easily accessed. Figure 20 shows the entity relationship (ER) diagram for the web server database as described in Chapter 4.The *ranges*, *users* and *patient* tables were filled with data representing sample patients so there would be information available that the applications could use. The Java servlets and web service running on the server could access this database using the Java Database Connectivity (JDBC) API. This provides libraries and methods for Java programs to access and process database data using SQL queries.

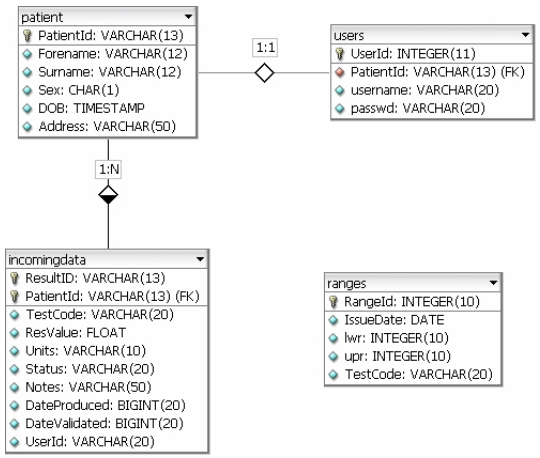

Figure 20: Web server database

5.3 Version One

This is the version of the system where the result is validated on the phone itself. It consists of a MIDlet client which runs on the mobile phone, Java Servlets running on an Apache Tomcat server (version 5.5.9) and the MySQL (version 5.0) database discussed in Chapter 3. The implementation of this version is discussed using sequence diagrams, based on the options available to the patient from the main menu.

5.3.1 Example Scenarios

There are four choices for the user from the main menu, Add and Validate a result, View and Upload Results, Delete Records and Update program. The sequences of events that a user may encounter using the system will be discussed.

5.3.1.1 Add and Validate a Result.

The sequence diagram for this can be seen in Figure 21. At the main menu, the first option the user has is to 'Add / Validate Result'. When the **commandAction()** method detects that the add result option is chosen the **genAddScr()** method is called. This method generates a new display, which contains a numeric text-box for the user to enter the result and a check to see if the user has been fasting. Upon selecting 'Add' the **commandAction()** method creates a new **ResultData** object using the result value entered, the current date and status. Using the **Date** objects **getTime()** method it sets a result ID which is a unique 13 digit number. Next the **addResult()** method is called. This method validates the result using the **ResultValidator.validate()** method, which checks to see if the result is within the normal reference range that it has stored in the record store. It then creates a byte array of the result information and stores it in the RMS data store. Next, the **genResDataScr()** method is called and the information about the new result is displayed to the user, they now have an option to upload the result or return to the list of results stored in the phone. The process of uploading the result, if chosen, is described in the next scenario. An example of screen flow of this scenario is shown in Figure 22.

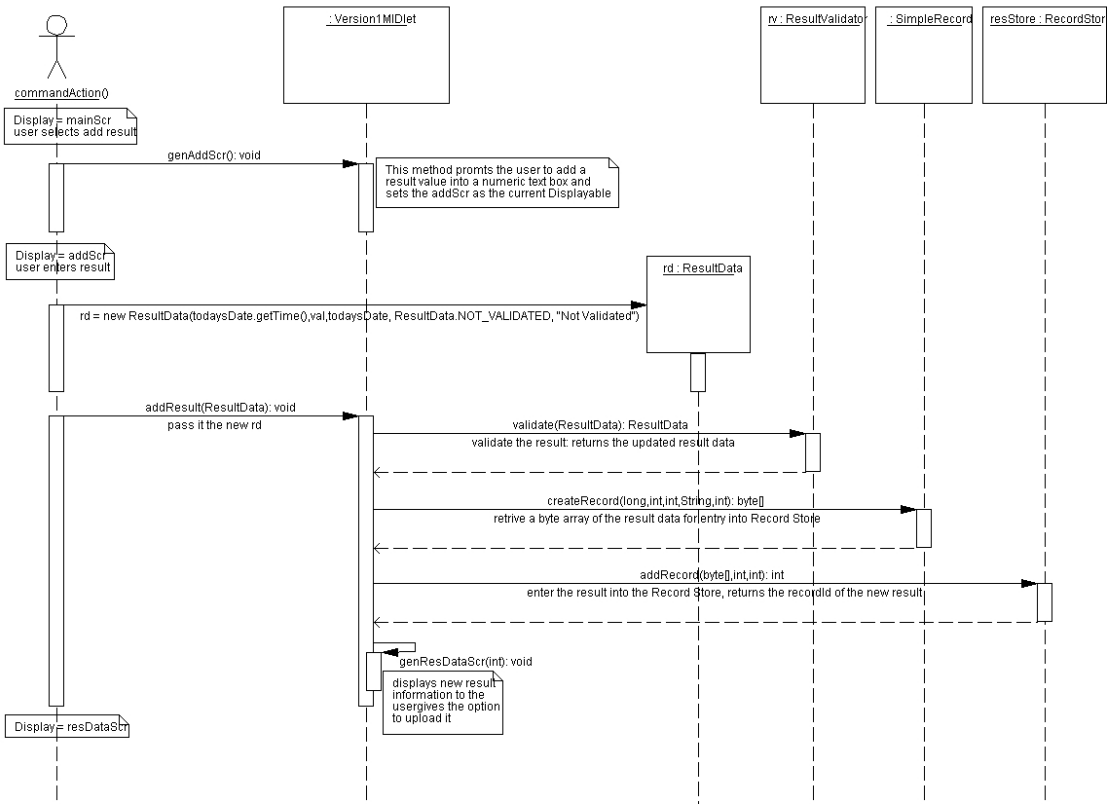

Figure 21: Sequence Diagram for Adding Result

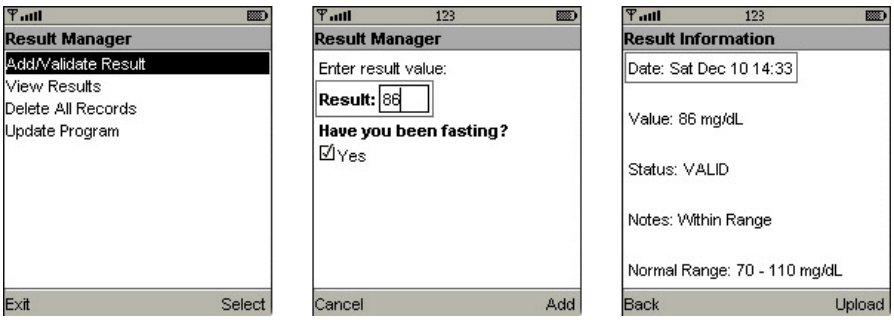

Figure 22: Screen Flow for Adding Result

5.3.1.2 View and Upload Results

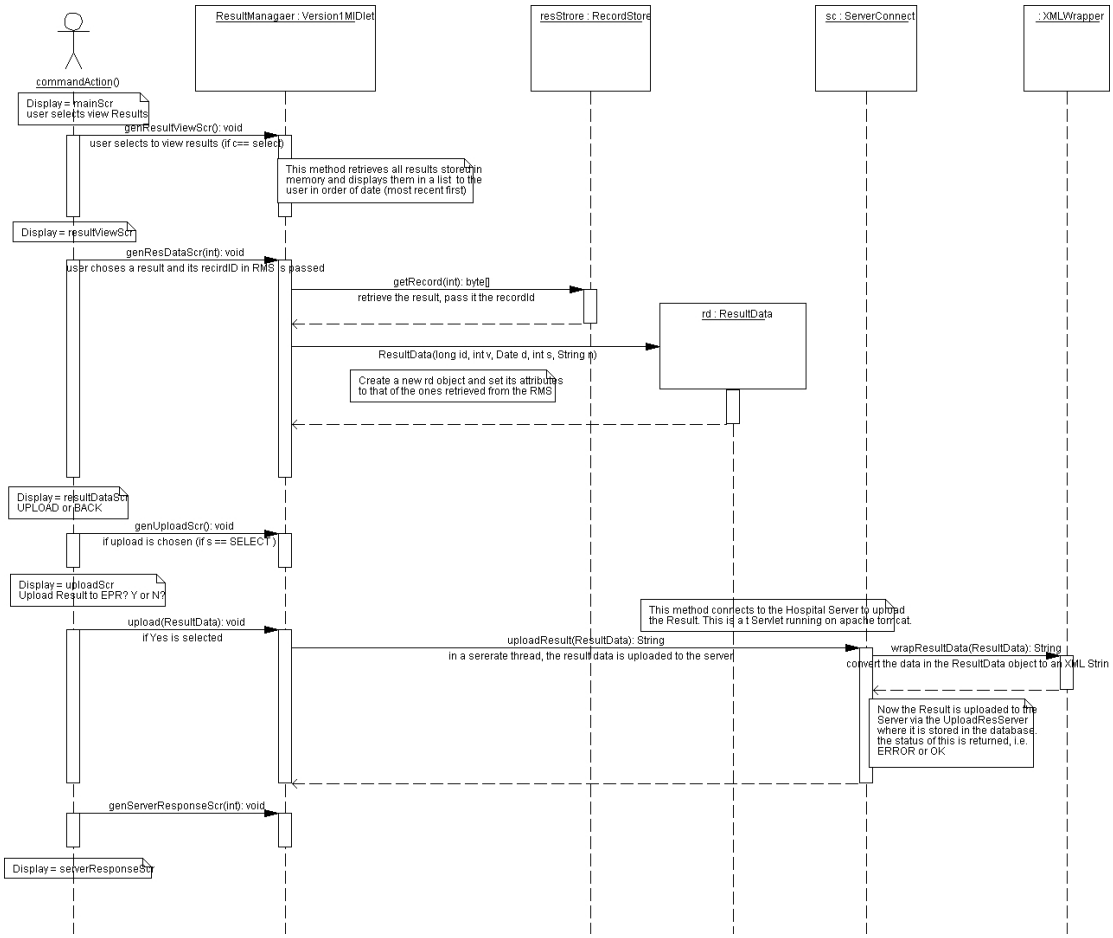

Figure 23: View and upload results sequence diagram

The second option for a user at the main menu is to 'View Results,' the sequence diagram for this is shown in Figure 23. When this option is selected the **commandAction()** function calls the **genResultViewScr()** method. All of the results are called from memory. These results are displayed in order of most recently added and

this **resultViewScr** is set as the current display. When the user selects one of these results its attributes are displayed to the user via the **genResultDataScr()** method. This method also sets the attributes of a new **ResultData** object using the result information retrieved from memory.

The user now has the option to upload the result to the server. If the choice to upload then the **genUploadScr()** method is called. This prompts the user again as to whether to upload the result to the server, in order to avoid connecting if 'upload' was selected by accident. If 'No' is selected the **genResultViewScr()** method is called and the user is returned to the result list. If the user confirms they want it uploaded, by selecting 'Yes', the **upload(ResultData)** method is called. This method starts a new thread, and calls the **uploadResult()** method from the **ServerConnect** class. It is good programming practice to perform all network activity in a separate thread as using the main system thread can result in the application freezing if the network does not respond [68]. So, to avoid deadlock, all networking code in the client applications is handled in separate threads. It also allows for the user interface to be active whilst there is network activity, which heightens the user's experience with the application.

```
<?xml version='1.0' encoding='utf-8'?>
<test_result>
 <result_data>
  <result_id>1133531808281</result_id>
  <user_id>MD00977</user_id>
  <test_code>GLUC</test_code>
  <value>86</value>
  <date_produced>1133531808281</date_produced>
  <units>mg/dL</units>
  <date_validated>1133531808453</date_validated>
  <status>2</status>
  <notes>Within Range</notes>
 </result_data>
</test_result>
```

Listing 3: XML Result Data

A HTTP connection to the hospital server is set up using the HTTP POST request method. The result is wrapped into the XML message format shown in Listing 3 using the **XMLWrapper.wrapResultData()** method and transmitted to the server over the HTTP link via a **DataOutputStream**.

The sequence diagram for the Java Servlet running on the server is shown in Figure 24. The server reads the data stream from the client MIDlet. The information it is receiving is the patient's result data in the XML format shown in Listing 3. First, the Servlet must parse the XML data. This result data is then entered into the database and a response code is sent back to the client MIDlet. The response will indicate what has taken place and whether the data was entered in the database successfully or not. A message reflecting the outcome of the hospital communication is shown to the user as in the screen flow diagram of Figure 25.

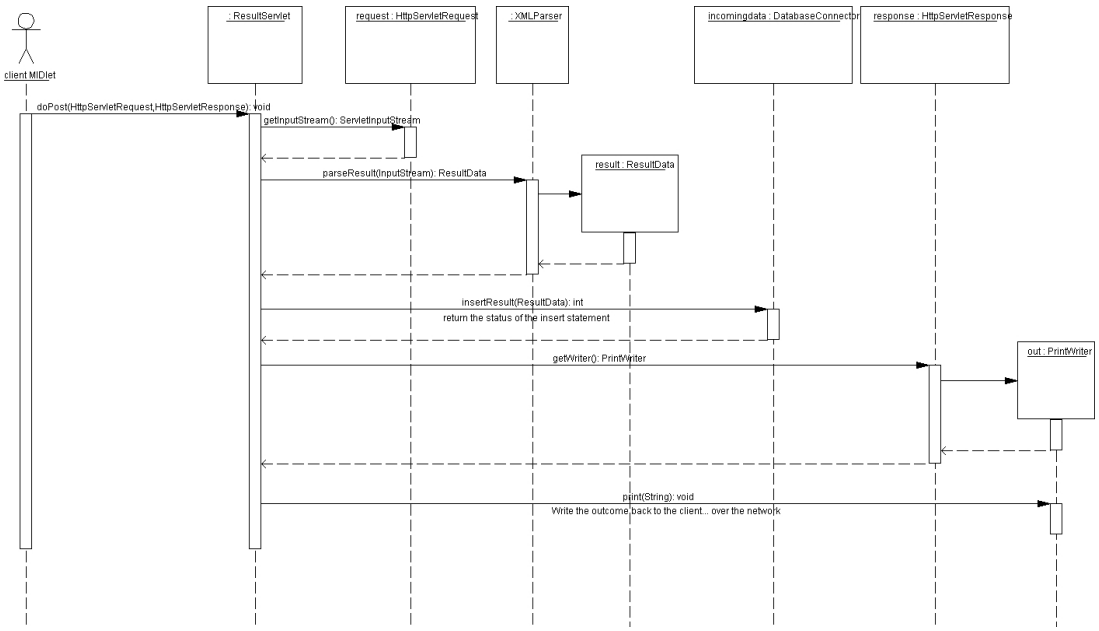

Figure 24: Result Servlet sequence diagram for version one

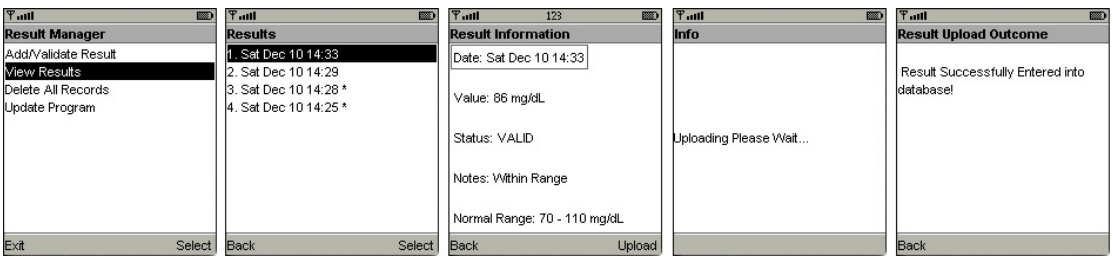

Figure 25: Screen flow diagram for viewing stored results and sending to hospital

5.3.1.3 Delete Records

The third function for the user is to delete results from the phone memory. This process was implemented in the same way for each version implementation and the sequence diagram is shown in Figure 27. Again starting from the main menu screen, the user now selects 'Delete All Records' which changes the display to a new screen asking the user to confirm the delete. If 'Yes' is chosen, the **`delRecords()`** method deletes the record store and displays an alert to that effect before returning the program to the main screen.

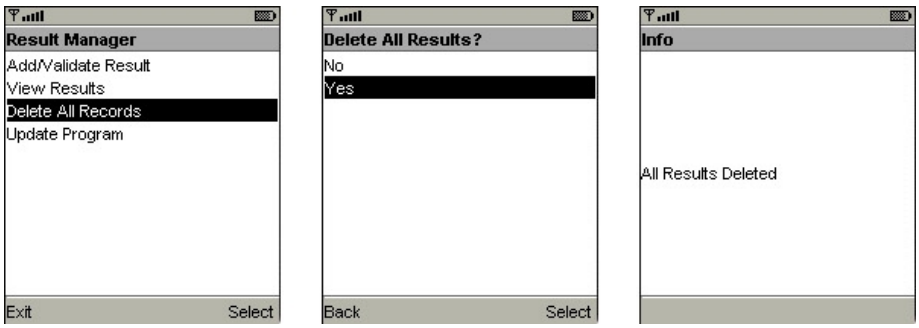

Figure 26: Delete records screen flow

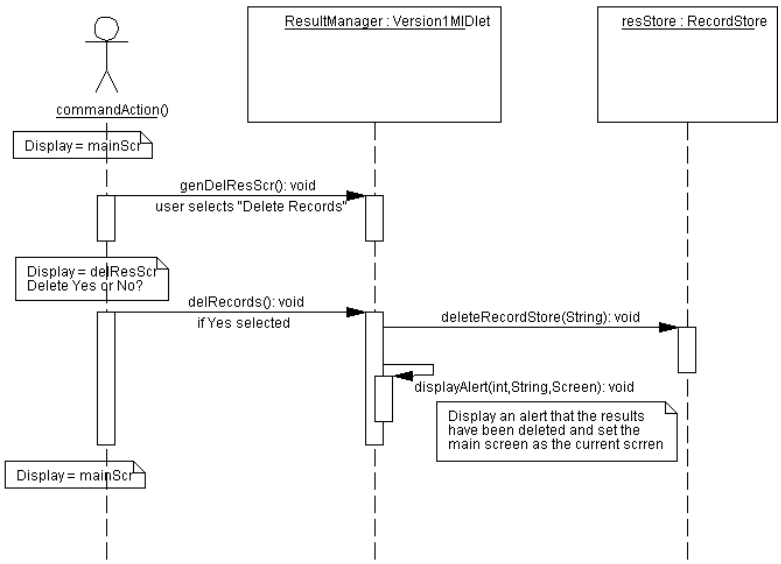

Figure 27: Delete records sequence diagram

5.3.1.4 Update Program

To demonstrate an example of this, a system for downloading reference ranges is included in this implementation. In the server database, the table *ranges* contains the latest reference ranges for particular tests, identified by the *TestCode* attribute. The client MIDlet can connect to a Servlet on the web server, which will query the database and return the latest ranges to the client. The client keeps a record of the ranges for the validation method in the phones memory. Figure 29 and Figure 30 show the client and server applications sequence diagrams respectively. When the user chooses to 'Update Program', the MIDlet application connects to the **`UpdateServlet`** on the server, which returns the ranges. Next, the MIDlet stores these ranges in memory using the **`RecordStore.addRecord()`** function before generating the main menu again.

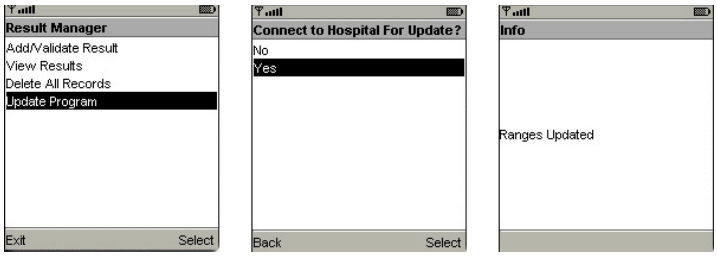

Figure 28: Screen flow for update

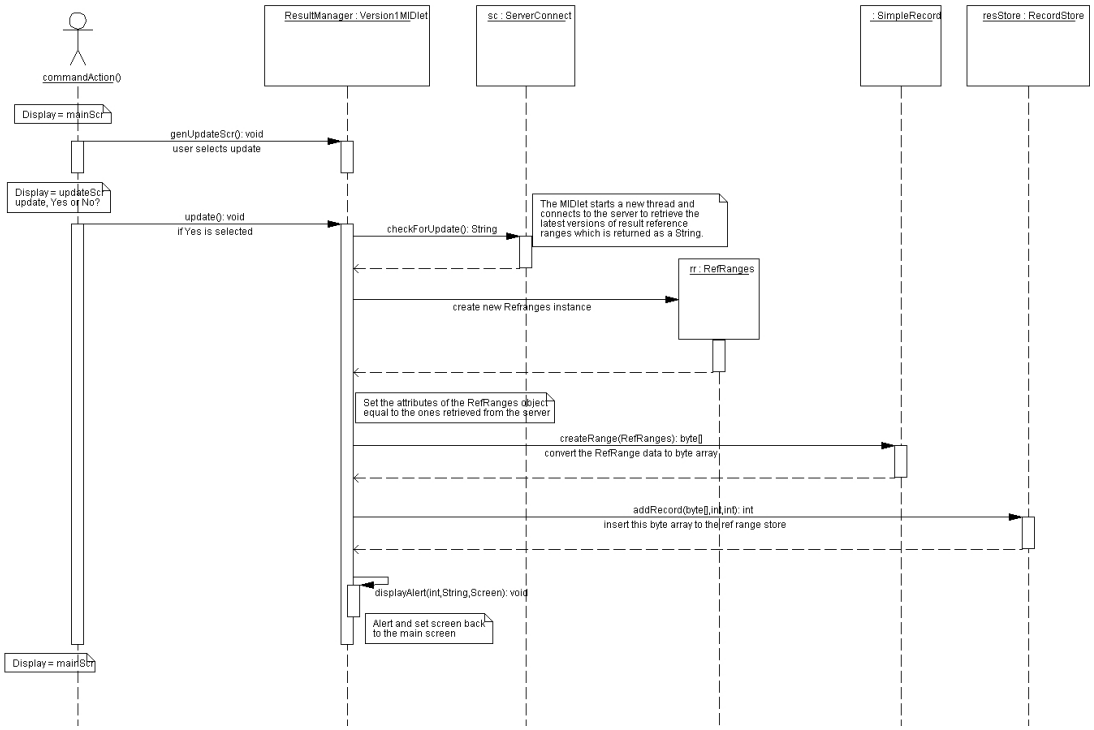

Figure 29: Sequence diagram for MIDlet updating reference ranges

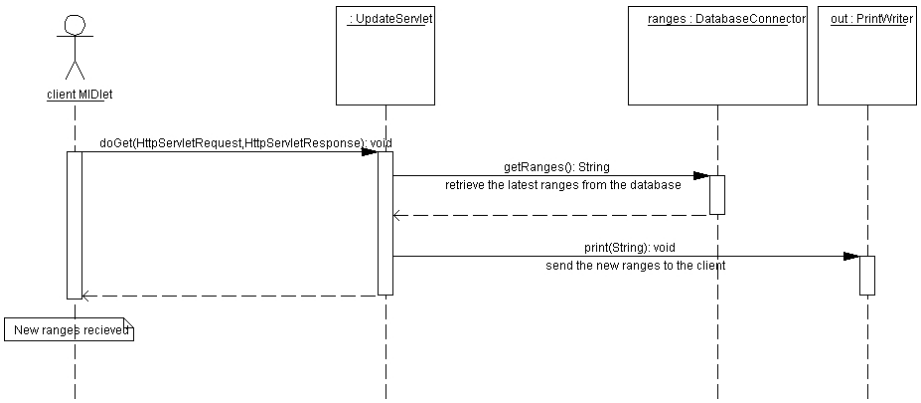

Figure 30: Sequence diagram for update Servlet

5.4 Version Two

The second version of the system requires longer network connections as the patient's test result is sent to the hospital for validation on the server and thus a longer wait is needed. Like the first version the user has the option to add and store a result in their phone, however it will also be uploaded immediately to the hospital for validation and storage in their EPR. As the validation happens on the hospital server it is possible to have a much more complex algorithm to validate the result and it can draw information from many more sources. Like the last section, the implementation will be described with sequence diagrams by going through the different choices from the main menu. Only the first two scenarios will be discussed in detail as the delete function is exactly the same as in version one.

5.4.1 Example Scenarios

The 'View Results' and 'Delete Records' functions are implemented in the same way as version one and hence do not need to be discussed in detail again. However, adding a new result and uploading to the server will now be discussed in detail.

5.4.1.1 Add and Validate Result

When the patient selects to add a new result, an entry form is generated similar to that in version one. The patient enters the result and presses 'Send' which invokes the **`addResult()`** method. Figure 31 is the sequence diagram for the client MIDlet. First the

result is stored in the phones memory, next it is wrapped in XML format as shown in Listing 3 and finally it is uploaded to the server. Again, as in the previous version, network activity is handled in a separate thread. A graphic indicating that the result is being sent is displayed to the user in the main system thread using the methods in the **ServerWait** class. When the Servlet, shown in Figure 32, receives the result data, it stores it in the *incomingdata* table in the database. Now it waits until the result has been validated, with the **retrieveWhenValid()** method. If it is not validated within a certain time it returns a timeout error to the client.

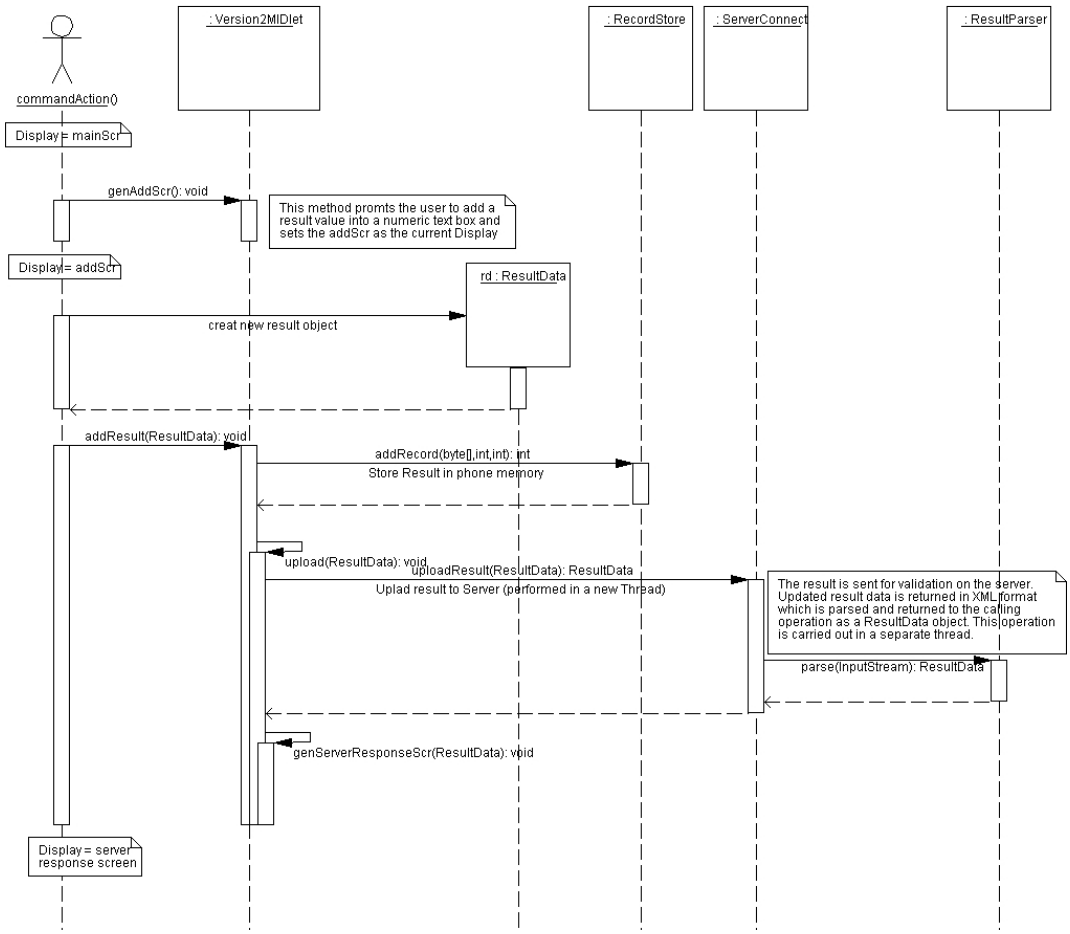

Figure 31: Sequence diagram for adding a new result

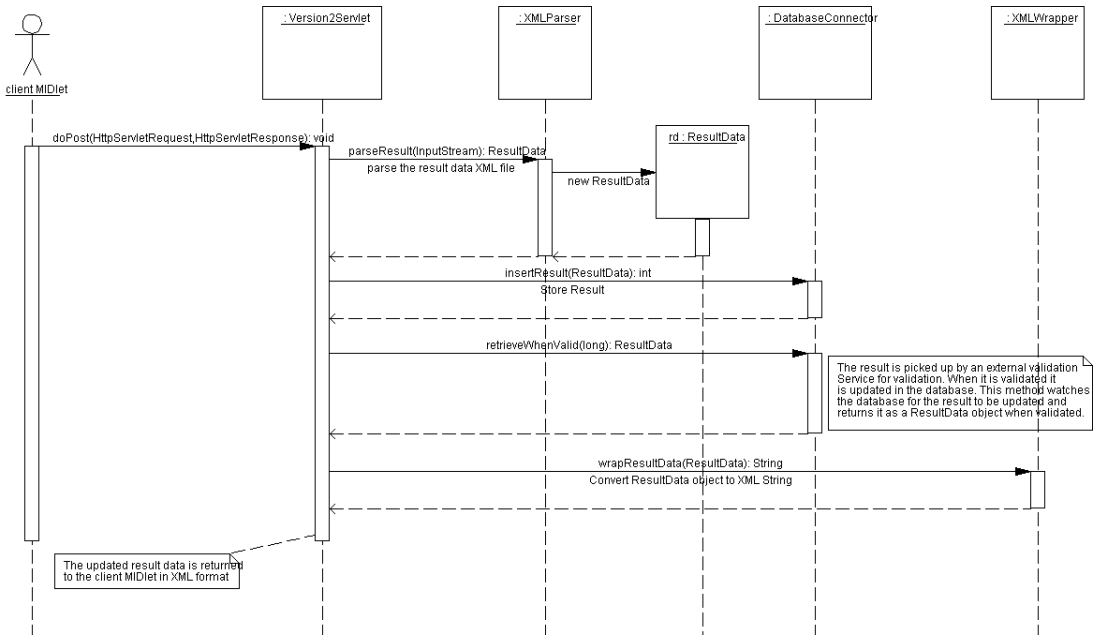

Figure 32: Java Servlet sequence diagram for version 2

The validation service runs simultaneously in the background checking for new results arriving. When it detects a new result it validates it. For this implementation, the validation algorithm is simple and compares the current result value to the patient's previous result value and confirms that it is within a specific range of values. When it has validated the result it updates the details in the database and moves onto the next result to validate. The sequence diagram for this is shown in Figure 34.

Figure 33: Screen flow for Add/Upload Result

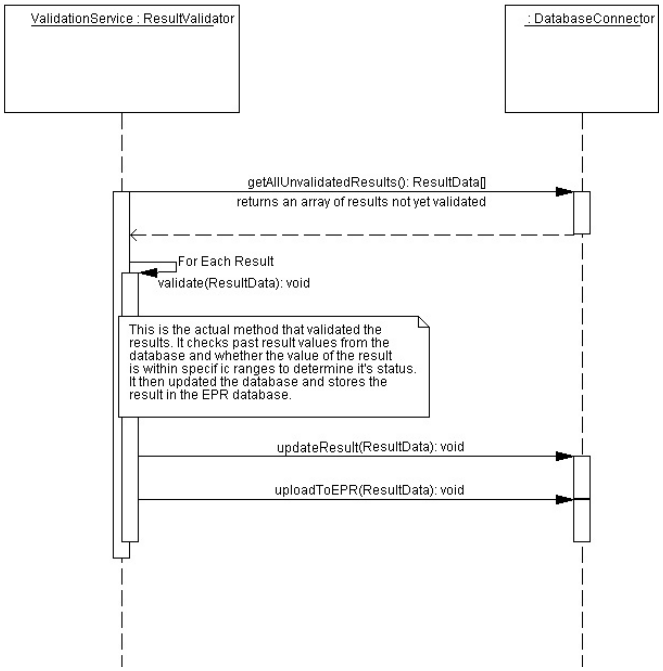

Figure 34: Validation service sequence diagram

When the result has been updated, the **retrieveWhenValid()** method, which is waiting for the result to be validated, returns the result details to the Servlet. The updated details are re-wrapped as an XML document and this is sent back to the calling client. The client application parses the document using the kXML parser. This is a third party package for XML document processing on J2ME devices. Once the information is extracted from the XML document, it updates it's record of the result and displays the outcome to the patient. The client application screen flow is shown in Figure 33.

5.5 Version Three

To a patient using this system, version three will appear to be the very same as version two. This version also validates and stores the patient's test results on the hospital server and then returns the outcome and advice based on that validation. However, it employs different technologies to send the result data to the server, namely web services and remote procedure calling (RPC), which form a distributed computing solution. The web service was implemented and deployed on the server as discussed in the previous chapter. The **Validate** interface defines how the client and server communicate and the methods on the web service that clients can invoke. In this implementation the web service contains one method, **validate()**. The web service interface was implemented in

the **ValidateImpl** class. It was compiled, built and deployed to the server using the tools available in Sun's Netbeans integrated development environment (IDE). This process also creates the web service descriptor (WSDL) file. When the service was up and running, the WSDL file was inputted to the J2ME wireless development toolkit and the stub code was generated for use with the client MIDlet.

5.5.1 Example Scenarios

As with version 2 the 'View Results' and 'Delete Records' functions are implemented in the same way as version one. The process for communicating with the hospital server using RPC to validate the result and the implementations will be discussed further.

5.5.1.1 Add and Validate Result

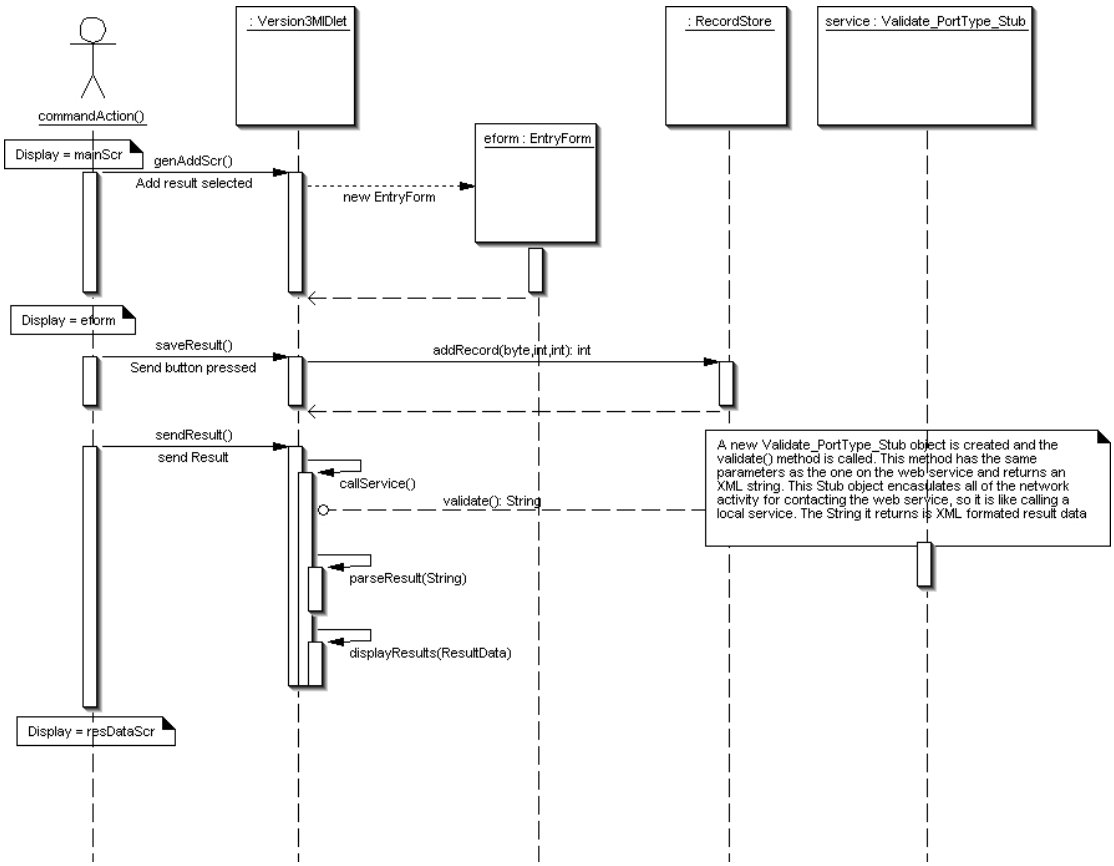

Figure 35: Sequence diagram for version 3 client application

This client MIDlet contains many functions similar to those in the previously discussed applications. The screen to add a result is generated in a different way to the previous

two versions. In this version a separate class called **EntryForm** was developed for taking user input. When called from the main menu, via the **commandAction()** method, the entry form is displayed. It also is different, as it requires the user to input a username and password for authentication on the system. The sequence diagram for this application is shown in Figure 35. When the patient has entered all of the data, the 'Send' button is pressed, which invokes the **saveResult()** and **sendResult()** methods. The **sendResult()** method formats the result data and then invokes **callService()**. A new thread is started as calling the service will involve network activity and an instance of the WSDL generated stub class, **com.mobilevalidator.validationservice. Validate_PortType_Stub**, is created. This object contains the **validate()** method which has the same architecture of the web service method. The application invokes this local method passing it the user ID, result ID, result value and a Boolean indicating if the patient has fasted. The communication is handled over a SOAP HTTP connection as discussed in Chapter 3. But how the method and its arguments are encoded and sent, and how the response is received and decoded are all transparent to the application. The actual logic to make the call to the service on the network is encapsulated in the **Validate_PortType_Stub** class and the other stub classes in the auto-generated package. The String value returned from the **validate()** method is an XML document containing result data similar to that of Listing 3. This is parsed and the record of the result in the phone memory is updated. Finally, The outcome of the validation and the notes are displayed to the user.

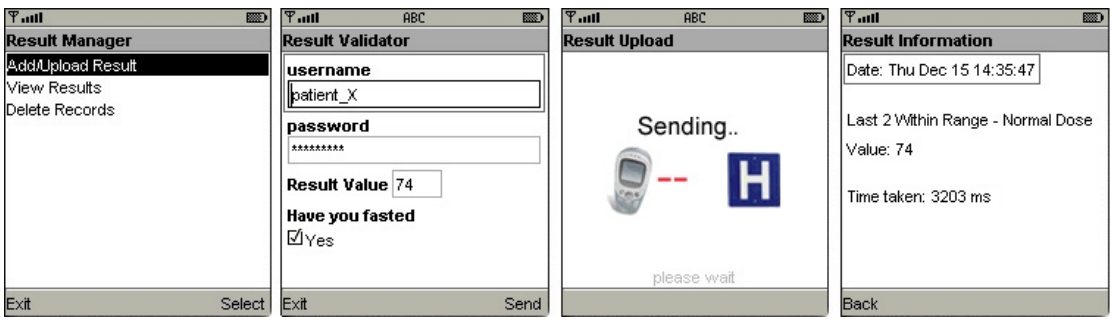

Figure 36: Version 3 screen flow

The implementation of the validation web service and the **validate()** method is shown in Figure 37. When the method is invoked, it stores the result in the database. An identical validation service to that described in version two, Figure 34, is running

simultaneously on the server. When it has validated the result the **retrieveWhenValid()** method sees it and returns it as a **ResultData** object. Finally, an XML document of the result data is built. This is converted to a string and returned to the calling method (the J2ME client in this case).

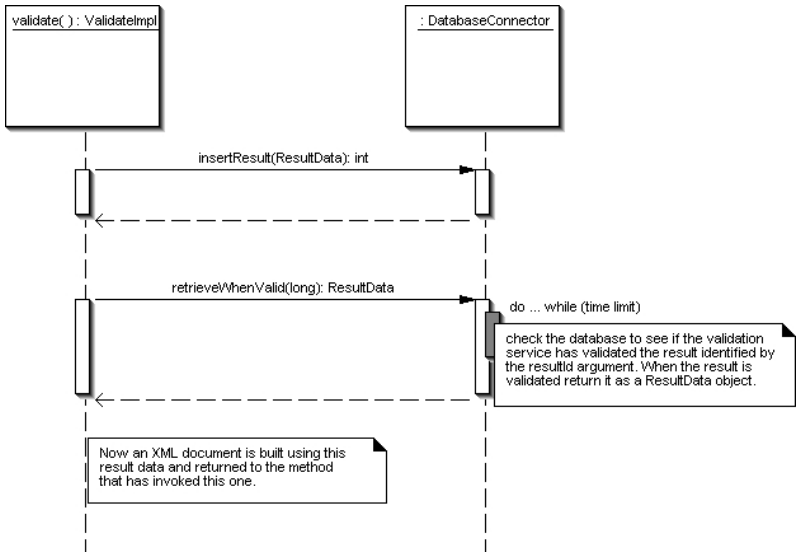

Figure 37: Validation service implementation

5.6 Testing the Applications

All three implementations were tested using the database and server tools previously described. An Apache tomcat server was set up and run on a computer running Microsoft Windows 2000 and the MySQL database server was implemented and run on the same machine. All of the server programs discussed were deployed on the server so that they were accessible by client applications over a HTTP connection.

A timing function was introduced to all three versions to record the time the particular application spent connected to the network. This was done to determine which version was the quickest and most responsive. A long network connection would make the application undesirable for a patient to use, as they would experience a lengthy wait for a validation or result upload. The timer runs from the moment the user presses the 'Send' key until the XML document has been received from the server, parsed and the result updated in the phone memory.

5.6.1 Phone Emulator Tests

The client MIDlets were tested on the Sun's J2ME Wireless Toolkit and some vendor specific phone emulators from companies such as Sony Ericsson, Nokia and Siemens. The vendor emulators are designed to run java applications exactly as the actual hardware would. With the exception of the third version using the additional Web Services API package (JSR-172), the three systems were designed using only classes from the CLDC 1.1 and MIDP 2.0 libraries to ensure platform independence and compatibility with a large range of devices. These libraries are the minimum requirement for any phone that is J2ME compliant.

Each version was tested to check that all of the functions implemented in it were working as designed. All of the programs were seen to run as desired on the different phone types, confirming that they were platform independent. As the mobile phone emulators run on a desktop computer and connect to the server over a fixed wire network the connection timer that was implemented was returning very low values. The network calls were made 100 times by the timing function and the average times recorded are shown in the table below.

Version	Average Connection Time (ms)
1	322
2	816
3	854

Table 9: Network connection times with emulators

These values were unrealistically low for a real life implementation that communicates over a mobile network. However they do indicate that the first version was quicker. This is because it does not have to wait for validation of the result. The time difference between version 2 and 3 is insignificantly small indicating that the validation and processing time on the server is the same for each one.

Testing on the emulators proved to be very useful for checking functionality and finding faults and bugs in the development of the implementations. However, to get an idea of how the system would really work, the applications needed to be tested on real mobile devices working on an actual mobile network.

5.6.2 Phone Tests

To test the applications in a more realistic situation, the applications were loaded onto a Siemens C65 mobile phone. This phone was used to test the first and second version implementations but could not be used to test version three, as it does not support the J2ME Web Services API. The applications were seen to work as expected with no difference to the emulator tests. When connecting to the server however they took longer, as expected. The average wait times experienced are shown below.

Version	Average Connection Time (ms)
1	3234
2	3988

Table 10: Network connection times on a real system

So using the application in a real system has added about 3 seconds to the network connection time of the emulated system.

5.7 Summary and Conclusion

The implementations of the three system designs outlined in Chapter 4 were successfully carried out in both a mobile phone emulator environment and on a real device. This chapter has provided a detailed description of the implementations of both the client and server applications for all three versions of the system. The sequence diagrams show the flow of each function available to the patient and how the methods and classes related to each other. The screen flow diagrams in this chapter are taken from the tests that were done using the J2ME wireless toolkit phone emulator.

The implementations proved that a mobile phone solution to the issue of connecting patients to hospital for validation of results is a viable solution. The average time taken for a user to receive advice back from the hospital in version 2 was less than four seconds. This is a very short time to wait and if it were doubled, for example, the system would still appear to work quickly. This means that much more logic and data processing could be implemented on the server side if necessary, as in a real hospital server. The connection times from both the emulator and real system indicate that most of the 4 seconds or so in the real system is actually spent transmitting and receiving the

data. This means that a four to six seconds delay on the server, for the purpose of ensuring a better service, would be tolerable. After performing theses tests it has been shown that all three versions are viable solutions and the speed of the service would mainly depend on the complexity of the validation algorithm.

6 Conclusions and Suggested Future Work

6.1 Conclusions of Research

This thesis has successfully reported on the current status in the telemedicine and medical informatics arenas, with regard to patient self-testing, the advantages afforded and the issues and problems that can result. Initially, thorough research was carried out in these areas to gain solid background knowledge of the areas of research. This included researching the current information and communication systems used in hospitals, clinical laboratory sample testing and validation of the results produced. Validation of results is a very important part of the laboratory QA process and is performed routinely in all automated clinical laboratories. An implementation of the INCA system, discussed in Chapter 3, was carried out in the early stages of the research to gain insight into how the test request and result reporting process works in a hospital laboratory environment. From this implementation, invaluable knowledge of distributed computing and laboratory information systems was obtained. The main problems associated with patients who self-test at home were identified. Patients who use POCT at home to manage their condition are isolated from the hospital that is responsible for their treatment in terms of their day-to-day care, and their results are not subjected to any validation process. Providing patients with a system of result validation and transmitting their results to the hospital for storage and monitoring would greatly improve the quality of patient care in the home health care environment. The work presented in this thesis can act as a model for implementing such a system.

In addition to discussing the medical areas of this research project, current mobile communications and software systems were also reported on. This is a rapidly growing area of ICT and using this technology for developing a solution to the problems at hand made sense. In addition, the ubiquity of mobile phones today means the service would be accessible by many people and no additional communications devices would need to be purchased. Use of a mobile phone to send result data to the hospital means the patient is not tied down and can avail of the service away from the home, whilst at work or on holiday. Open source technologies were used in this research including the Java platform, Apache servers and MySQL database management systems. The micro

edition of Java (J2ME) provided a powerful and feature rich platform to program the mobile applications. Most phones available today support Java technology.

After the background and review of the technologies was discussed, the three alternative designs of a system for providing patients with remote healthcare assistance were introduced. A full discussion of each of these designs and the methods and tools to be used with them was given in Chapter 4. These systems provide a useful framework for development of a full system. Each version provides a feature that makes it unique from the other two, such as version one validating the result on the phone. This meant that different features could be concentrated on when trying to evaluate the optimum system, rather that designing one very complex solution. All three designs consist of a client mobile phone application for the patient to enter their test result values and transmit them to a web server in the hospital. Applications running on the web server process the results and store them in a database so they are available for access by the validation service and other services if necessary before being sent to the EPR.

A proof of concept for each version was implemented. All of the applications were written using the Java platform. Initial tests during the development were carried out using a number of mobile device emulators from different mobile phone vendors. This helped to verify that the system would work on a range of different devices and device types. It was seen that the applications worked well on all types of phone tested but they looked better and were much clearer when emulated on phones with larger displays. Once the implementations were working as designed on the emulators they were installed on an actual device, which was capable of running Java applications and making GPRS connections. The Apache Tomcat server that contained the server applications was opened so it could be accessed from any client outside the college firewall.

Testing the applications on a real device proved that the system could work well in a real production environment. The time to transmit a result, process it on the server and receive and process the response on the phone takes only four seconds or less as the amount of data to transmit each way is in the region of only a few bytes. This time could be increased to more than double before the system will appear to operate too slowly. Hence, more server application logic or a busier hospital server should not

interfere with the overall operation of the current system. The first version of the system design is capable of validating a result on the phone itself and can connect to the server to retrieve updated validation information. Using this information combined with the mathematical and data storage libraries available in J2ME meant relatively complex result validation was possible on the mobile phone. The first version had the quickest turnaround time when uploading the result to the server as it doesn't have to wait on a validation. The second and third versions allowed the result to be validated by the actual hospital validation system, ensuring the results were validated with the gold standard process. Both versions worked equally well in terms of turnaround time. However as web services is a growing area of distributed computing and is being supported by more and more clients, version three would make the most sense for any further implementations. The user interface on all three versions is simple to use, with only a few key menu choices. Thus, the applications could be used by any patient, who is already capable of operating a POCT device, after only a very brief introduction.

This thesis encompasses a comprehensive review of the state-of-the-art in the telemedicine, medical informatics and ICT fields and based on this review a system for patient-hospital communication has been designed and a proof of concept implemented. There is no doubt that the system developed in this thesis provides a good blueprint for anyone who wants to fully implement a similar system. The tests carried out prove that such a system has great potential for improving the quality of patient care. In addition, the research carried out for this thesis adds to the body of knowledge of how mobile communications can be used to improve patient healthcare without dramatic increases in costs.

6.2 Suggested Improvements and Future Work

The system designed for this thesis met the required specifications and worked well when implemented. It provides a very good base for further developing the system into a fully featured remote validation service for use in the real world. Implementation of a fully working system would necessitate a much longer time period to the time limit of the research. However, from the work developed, important issues have been identified which should be addressed and many modifications could be instigated to improve it.

6.2.1 Device Connectivity

In order to eliminate erroneous result data being entered into the client application, there should be a system implemented to transmit the information from the POCT device to the mobile phone. This could be done over an IrDA link or using a Bluetooth connection [12]. J2ME contains additional packages for Bluetooth connectivity so it would be possible to introduce interfacing with a Bluetooth enabled POCT device to the current applications. Implementing such a feature would greatly reduce the chance of human error occurring and thus make the existing system more robust and reliable.

6.2.2 Security and Encryption

The system designed in this thesis keeps patient information private by using usernames so it does not transmit any actual patient identification data over the network. The server can then log the patient in and retrieve actual patient information, if necessary, using the user details. This is the first step in ensuring the privacy and anonymity of the patient. However, patient privacy and confidentiality is one of the most important issues in a hospital system and introducing a system that puts patient data in a vulnerable position is not acceptable. Information that is transmitted over a wireless link is subject to interception by unauthorised parties. A number of steps may be introduced in order to ensure this system protects patient information.

The first step is to configure Tomcat to be a secure server. Next, ensure all network communication is carried out over a secure connection. J2ME is capable of making Secure HTTP (HTTPS) connections using the Secure Sockets Layer (SSL) [69]. The information should be encrypted before being sent so that it is transmitted as a seemingly meaningless stream of data and then decrypted by the receiver. Finally the integrity of the data should be verified to ensure it has not been altered in transit. Implementing these methods will ensure a far more secure and reliable system that patients will be quicker to trust.

6.2.3 Display Size

Although the applications developed have worked and functioned as desired when testing, one point that was noticed when testing the system on the Siemens device was the cluttered appearance of some information on the screen. The Siemens device had a

screen size of 130 by 130 pixels, which is relatively small compared to many of the phones available today. For the applications to present the data in a neater and more presentable form, a larger screen is recommended. The methods for displaying result data could be reviewed and modified to arrange the display data in a more compact way, but using the application on a phone with a medium to large display size would ensure the information is presented in a clear and easy to read manner.

6.2.4 Alternative Client Types

The server programs implemented are not specifically designed for use with mobile phone clients. This means that a wide range of client types could be developed to avail of the service, including a stand-alone PC application, a HTML web browser system or a PDA with internet access. To broaden the scope of this system, different client types can now be developed to interface with the existing server applications, thus providing the patient with the same services. This could help the system reach a wider range of people who may not be comfortable with a mobile phone solution.

Appendix A: Nomenclature

2G	The second generation digital GSM mobile network
3G	Third Generation mobile communications standard capable of data rates up to 2 Mbps.
API	Application Programming Interface. A library of java classes and the available methods within the classes.
CLDC	Connected Limited Device Configuration. A set of API's for developing Java programs on mobile devices with limited memory.
CORBA	Common Object Request Broker Architecture. Enables communication between distributed objects.
EPR	Electronic Patient Record. A secure, well-formed and electronic method for storing patient information in a hospital system.
GPRS	General Packet Radio Service. Sometimes referred to as 2.5G, provides increased data transfer rates in 2G networks.
GSM	Global System for Mobile Communications.
HIS	Hospital Information System. For storing and handling hospital data and information.
HL7	Health Level Seven. An American National Standards Institute accredited standard for hospital data communication.
HTTP	HyperText Transfer Protocol. This is a protocol that allows the exchange of data between clients and servers. Primarily used for the exchange of HTML documents on the internet it can be used to transfer a variety of data types.
HTML	HyperText Markup Language. A standard way in which a document should be structured so it can be viewed universally on the web. A set of tags define how a document's content should be interpreted.
INCA	Integrated Networked Clinical Analyser. A standard for linking clinical analysers to Laboratory Information Systems.
INR	International Normalised Ratio.
J2EE	Java 2 Platform Enterprise Edition. For server programs and web services.
J2ME	Java 2 Platform Micro Edition. Designed for resource constrained

	mobile devices.
J2SE	Java 2 Platform Standard Edition. The standard, desktop edition of the Java platform.
JAX-RPC	Java API for XML-based Remote Procedure Calling.
JAXP	Java API for XML Processing.
JRE	Java Runtime Environment. Contains all of the basic API's needed to execute Java programs on a computer.
JVM	Java Virtual Machine. For executing the Java byte code.
KVM	Kilo Virtual Machine. A JVM optimised or resource constrained devices such as mobile phones.
LIS	Laboratory Information System. Responsible for handling clinical laboratory data and processes.
MIDP	Mobile Information Device Profile. A set of API's which allow J2ME developers deal with mobile device specific issues such as networking and record management.
POCT	Point of Care Testing. Clinical testing at or near to the place of patient care.
RDBMS	Relational Database Management System. A system where data is stored in a collection of tables related to each other through common data values.
SMS	Short Messaging Service. Simple text-based mobile communication standard.
SQL	Structured Query Language. A language for accessing and manipulating data in databases.
UMTS	Universal Mobile Telecommunications System. European standard for 3G development.
WAP	Wireless Application Protocol. Standard for linking wireless devices to Internet services.
WSDL	Web Services Definition Language. An XML-based protocol that is used to describe a remote web service.
XML	Extensible Mark-up Language. A customisable system for structuring and organising documents.

Appendix B: POCT Equipment

There are many, many manufacturers of Point of Care Test (POCT) Equipment for both home use by patients and for use at hospital bedside by trained staff. These Devices range in complexity, from small handheld devices to sophisticated laboratory style analysers. As well as the many manufacturers, the different types and categories of device is also very large. The test method used by one device is often different to that of another vendors' device of the same type. The functions available to a user or patient are also varied, some being very simple and merely producing a result to one test, and others being able to analyse a sample, produce many results, store them, profile the past results, connect to a LIS or computer and recommend medication dosages.

It was important for this research that the most popular functions used by these devices was found. A list was compiled, profiling the different aspects of the POCT Equipment. Some devices of American origin are categorised by the CLIA on their Complexity, if they are CLIA waived that means that they are easy enough to be used by a patient in the home. Some are 'Moderately Complex' which means they would be used by a professional like doctor in his surgery or in a clinic.

The Different Categories

Here are some of the main ones under the meter type category:

- Glucose Meters or Glucometers
- Blood Coagulation and Clotting units, Pothrombin Time, INR etc.
- Cholesterol
- Lactose Meters
- Holter Monitors, Event Recorders and ECG units
- Blood Pressure Monitors
- Sats Monitors (SpO2)
- Spirometers
- Blood Gas
- Urinalysis

Some of the devices of this list are shown over the next few pages.

Device Name	Device Type	Input Required	Value returned	Other Details
Bayer Ascensia® DEX® 2 Blood Glucose Monitoring System	Glucometer	Blood Drop,	Glucose Level (Numeric)	Saves information for up to 100 tests with dates, times and 4 daily averages. Download the meter's information to PC to manage data.
CardioChek Analyzer	Blood Analyser and Glucometer	Blood Drop on Test strip	Total cholesterol, HDL, triglycerides, glucose, ketone	Internal storage/review past measurements
CardioChek PA Analyzer	Blood Analyser and Glucometer	Blood Drop on Test strip	Total Cholesterol, HDL (good cholesterol), Triglycerides, LDL (calculated by analyzer), Glucose, Blood Ketone, Creatinine (only for medical professionals)	Internal storage/review past measurements
PrestigeIQ Glucose Kit (meter,10 strips,supplies)	Glucometer	Blood Drop on Test strip	Glucose Level (Numeric)	Date & Time, 14 & 30 Day Averaging
HealthFrontier ecg@Home	ECG Monitor. Bandwidth: 0.5 - 30 Hz linear phase	Thumbs placed on electrodes, one external lead to leg.	will acquire 10 seconds of Lead (I) and Lead (II), deviation of the ST segment of the wave, duration of the QRS complex, abnormalities of the T-wave.	The recorded data is sent to a data warehouse via Internet, wireless device, email, or via the built-in trans-telephonic coupler to tele-health center, where it can be immediately tracked, scanned and analyzed or sent to caregiver.

Device Name	Device Type	Input Required	Value returned	Other Details
HealthFrontier ecgAnywhere	ECG Monitor. Bandwidth: 0.05 - 75Hz (-3dB)	12 standard lead with simultaneous recording	simultaneous acquisition of 12 lead ECG tracing in 10 seconds.	record and store an ECG tracing and transmit it over the Internet via a PC, handheld PDA etc.to a remote receiving centre where it is stored and retrieved by the patient's care-giver. Bandwidth 0.05 - 75Hz (-3dB)
ICT (International Technidyne Corporation) ProTime prothrombin time tester	Prothrombin Time (PT) (Coagulation)	Blood Drop	Results are reported in both INR and PT seconds	Store 30 results in memory. The Cuvette uses three channels which test the PT and two on-board quality control tests, at the same time. The instrument and cuvette quality controls function together to ensure correct sample size, collection technique, test procedure, device functionality and reagent integrity
BD Logic™ Blood Glucose Monitor	Glucometer	Blood Drop	Glucose Level (Numeric)	Use a High/Low Blood Sugar Reference Card. warning if Blood Glucose Higher Than 600 mg/dL or lower than 20 mg/dL. Option to record Insulin volumes insulin type and dose for reference later. Can also obtain averages of test results.
CARDY Home ECG Device	ECG Monitor	6 electrode leads placed on body	ECG data is sent to the software on a PC and here it can be viewed and stored etc.The software can dtermine various parameters associated with ECG measurements.	Software for PC needed for carrying out traditional, laboratory ECG tasks: measurement, analysis, data storing, comparing. as well as programming CARDY® HOME, and downloading data from its memory.

Device Name	Device Type	Input Required	Value returned	Other Details
Welch Allyn's Spot Vital Signs	non-invasive blood pressure, pulse rate, temperature, mean arterial pressure and SpO2	Various: finger probe, arm cuff	All Numeric values of the measured signs.	all results in 30 seconds. external printer is available for record keeping purposes
Roche Diagnostics CoaguChek S	Coagulation meter	10 ul blood drop	INR (international normalized ratio)	advanced-generation reflectance photometer for on-the-spot measurement of prothrombin time. Results can then be relayed to Health Care Professional, who will check that they fit within personal target range and optimise warfarin dose. All necessary calibration information is provided by a code chip - no manual calibration is required. Internal quality control (QC).
Dráger Medical OxyTrend	Pulsed Oxymeter (SpO2)	Finger Probe	Pulse Rate, SpO2.	

Device Name	Device Type	Input Required	Value returned	Other Details
ITC Hgb Pro	Hemoglobin Test unit	Blod Drop on Test Strip	Hemoglobin	measures optical reflectance. Blood is applied to the test strip where it comes in contact with a red blood cell lysing agent which induces red blood cells to burst and release hemoglobin. It measures the change in reflected light before and after blood application to determine total hemoglobin.Result reported within 10 seconds of sample application.
Cholestech L·D·X® System	Blood Analyser and Glucometer	Blood Drop	Measures TC, HDL and triglycerides, calculates the TC/HDL ratio and estimates of LDL and VLDL.	Measures TC, HDL and triglycerides, calculates the TC/HDL ratio and estimates of LDL and VLDL. Factory calibrated
Micro DL Spirometer	Spirometer	Patient Blows Air through it	Date/Time, FEV1, FVC, PEF, FEV%, F50, F25, MEF, FET (All measurements printed as result, %predicted, predicted and normal range)	Stores Up to 500 complete data sets including the Flow/Volume & Volume/Time curve. Modem facility. Software: powerful trend packages available featuring the ability to overlay Flow/Volume curves from previous tests and spirometry data collection

Table 11: POCT Devices

Appendix C: INCA

This appendix is to describe the version of the INCA system that was implemented as part of the initial research. It was implemented using the MSc thesis of F. Knox [16], who originally developed the system.

Background to INCA

INCA is a system that was developed to provide a standard specification for a CORBA instrument interface in hospitals. This means analytical instruments can be connected to the hospital LIS so that results can be transferred automatically to clinicians and requests for specific tests on a sample by a particular instrument can be made.

Doctors can *request* specific tests from remote locations. These tests will be carried out on specimen samples that have been taken from a patient using the instruments in the lab, either automatically or with assistance of a laboratory technician. When the test result or results are ready, they are available for the doctor to view. In addition to the above, a doctor can *subscribe* to a specific set of results given input conditions, such as all test results from a certain patient or all test results of a certain type. There is also a validation client (or various ones), which take results and ensure that the values are valid subject to certain rules, which may involve using information from the patients record. This section of the system was not actually implemented in the original version of INCA but it was discussed and is an important part of this work, so a simple validation client was written in for this implementation.

Database

In the original specification, flat files are used to store all of the data passing through the system. This implementation utilises a database to store all information needed to run the INCA system. Some of this information is only temporarily stored and removed when no longer needed e.g. orders for results are removed when they have been obtained, and some is stored long term, such as patient details. A database was created

using Microsoft Access for the system. The programming language used is Java, which has imbedded SQL statements to manipulate the database.

A requirements analysis was performed first, which involved looking at the problem and writing out the requirements and tasks as plain English sentences so as to identify the entities and attributes so that tables could be designed for the database. Some of the tables were designed using the information about the main objects in the INCA document, so they could hold data relating to all of the attributes of the object. Some of the entities identified were *patients, samples, results, tests, specimen types, analysers and requests*. The database designed is shown below.

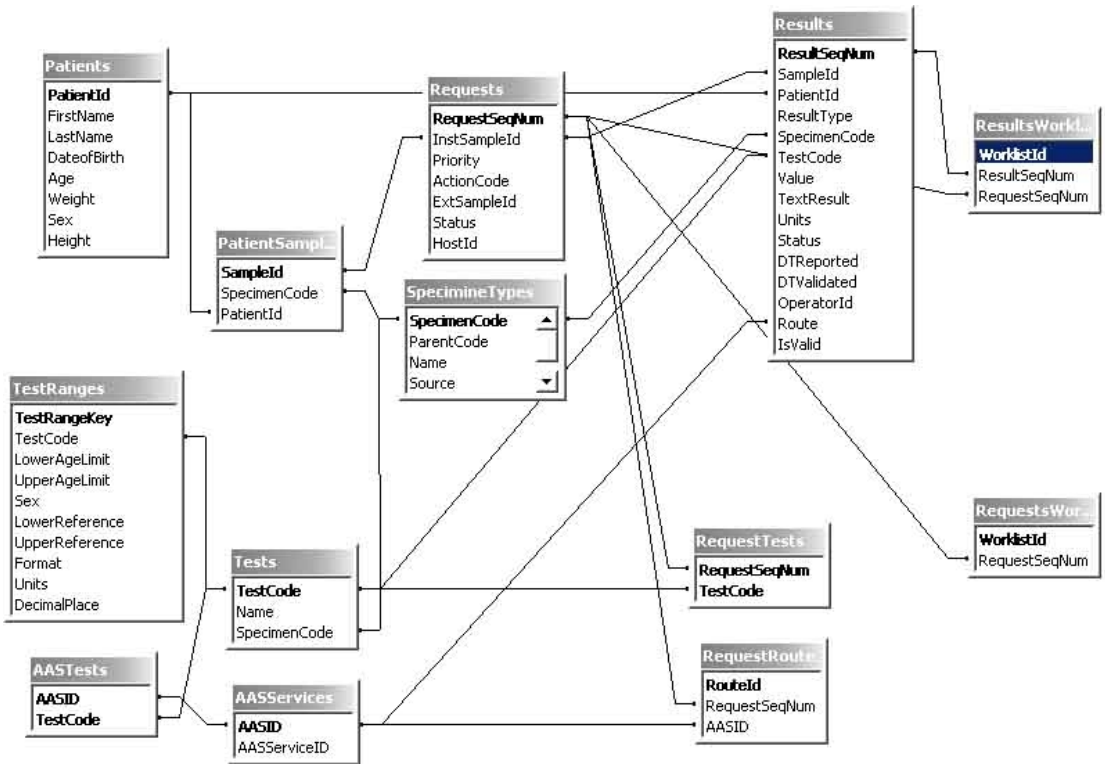

Figure 38: E-R diagram for INCA database

The INCA document had detailed descriptions of objects in the system, the *Results* and *Requests* tables were designed by just making them represent the corresponding objects. The *Worklist* tables were designed using information from the INCA document also. Information about specimen types and tests were obtained from a sample database for

the lab of St. James hospital, and the Tests, Test Ranges and Specimen Types tables were designed using it.

The Implemented INCA System

In the original implementation of INCA by Knox, C++ was the programming language used. As this system needs to be platform independent and is to be implemented on mobile devices, Java is a far better choice of implementation language. This means that all of the coding had to be redone using Java. As the original was built on distributed object technology using CORBA, the change to Java was not too difficult as it has good packages for CORBA programming.

To help describe the system a typical workflow will be used. When a *patient* is admitted to hospital they will have their details recorded and will be assigned a Patient Identification number. A doctor may order some *specimen* samples (such as blood or urine) to be taken from the patient so that certain *tests* relating to the patients condition can be performed on the sample. When the sample is taken it is labelled, recorded in the database and sent to the lab. Each specimen type is suitable for certain tests. The treating doctor or clinician can call up the patients' details and see what sample types have been taken, and choose one of these for tests.

The following is a rough outline of the sequence of events that occur once the samples have been taken.

1. The doctor enters the sample Id he is interested in having tests performed on. By looking up the database on the server, the system will show a list of the corresponding tests for the sample entered and the doctor can select the tests he wishes to have performed on it. This information is bundled into a *request* object and is entered into the database on the server via a CORBA connection. So now a request has been made and a request sequence number is generated and returned to the user. The request is entered into a work-list, which is a list of the requests for the analysers to process and is represented by a table in the database. The request is made using the methods available in the INCA IDL file.

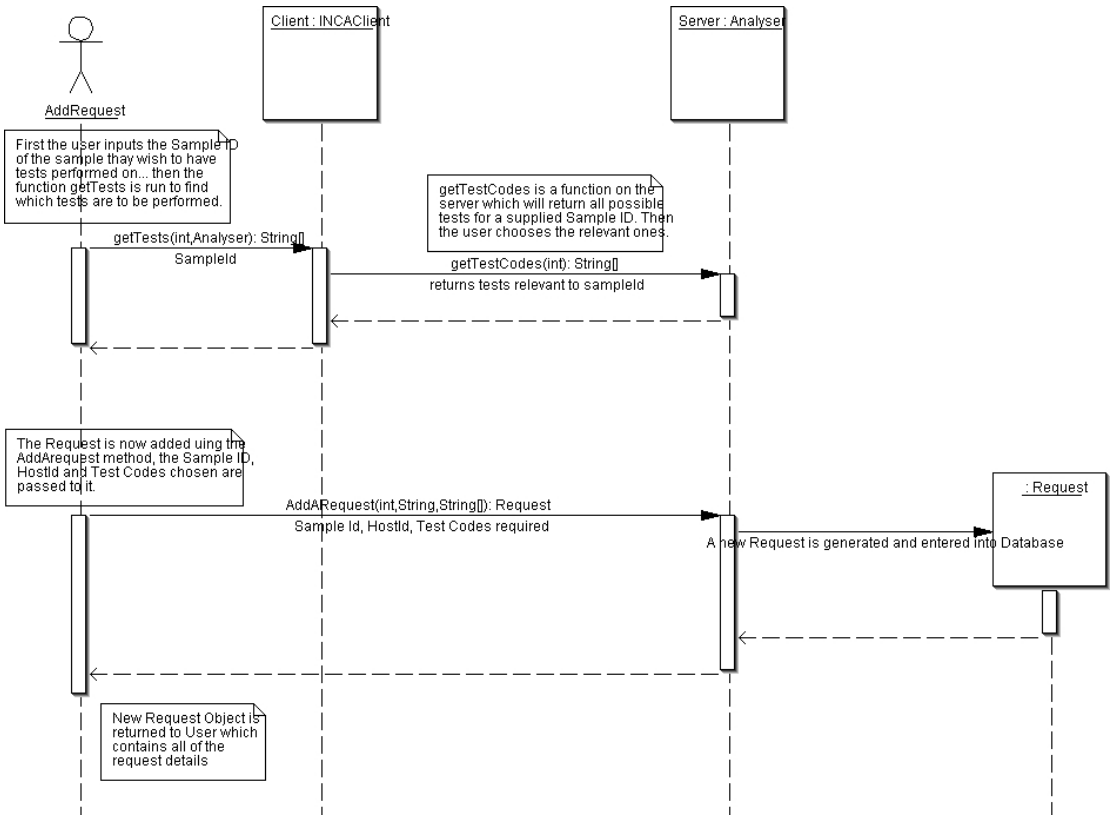

Figure 39: Add request sequence diagram

2. In the lab there are analysers, which are normally automated but may be operated by lab technicians. The *analyser* moves through the work-list and deals with the requests relevant to it in the order they were placed in the list. It verifies the sample ID in the request with the physical sample by reading the bar coded label. Next it performs the tests that were requested. For each result produced a new result object is constructed and placed in the database on the server via the CORBA interface. The result is added to the database using methods available in the Advanced Analyser Services (AAS) interface in the AAS IDL. The result is also added to a results work-list.

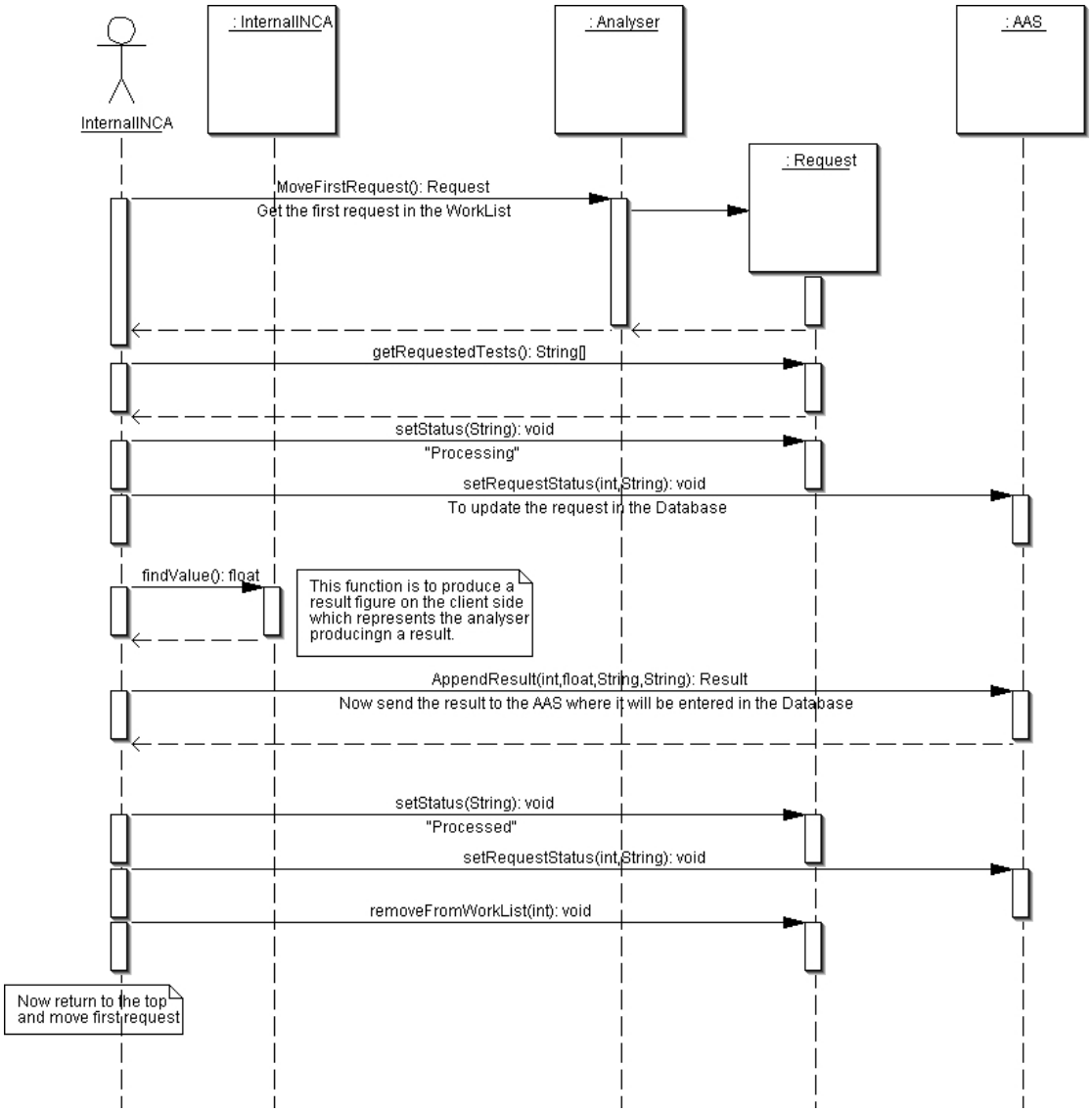

Figure 40: Internal INCA sequence diagram

3. Another client using the AAS methods is the validation client. Similar to the way the Analyser moved through the requests in the requests work-list to perform tests, it moves through the results work-list and validates the results queued. The validation of results is very important and is in simple terms a measure of the 'believability' of the result value and is assessed against internal consistency and against clinical information. Thre are various advanced validation checks used in the clinical environment [23]. In this implementation the validation service consists of a very simple check of the result value to see if it is within a certain range. If it is it is set as valid, if not it is deemed invalid. The status attribute for the result is updated to reflect whether it is valid or not and the database on the server is updated.

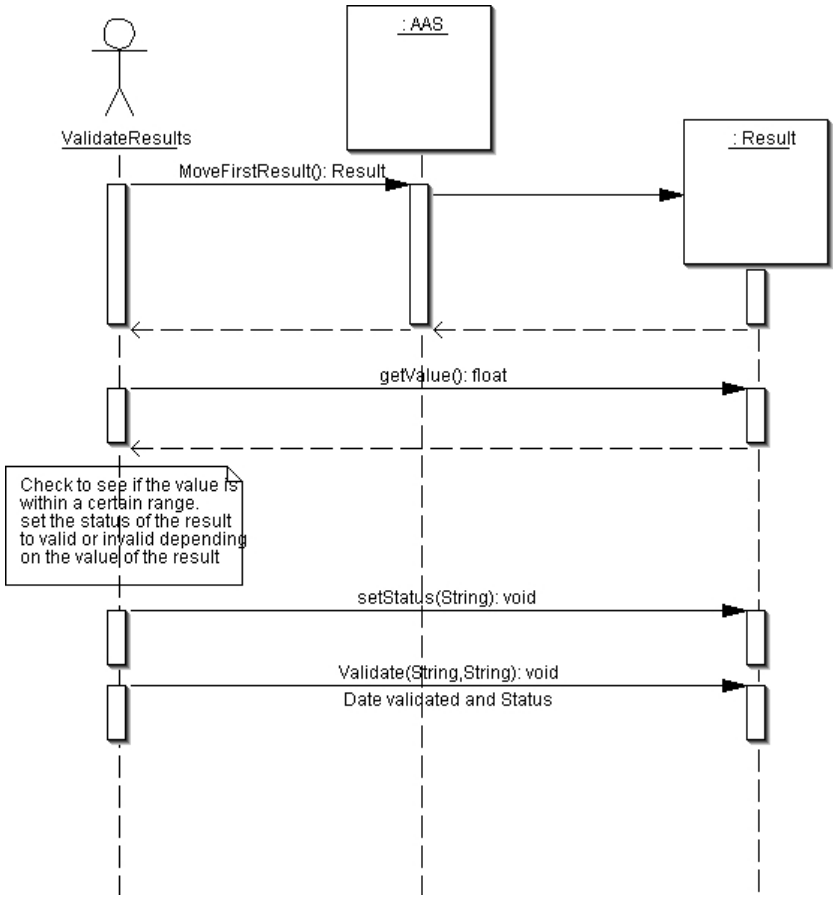

Figure 41: Validation sequence diagram on INCA system

4. The doctor who made the original request for tests in 1 above is able to check the status of requests s/he has made at any time using the request sequence number. If the sample has been analysed and results are available they will be displayed along with their status (valid, invalid or not yet validated). This is done using the AAS module.

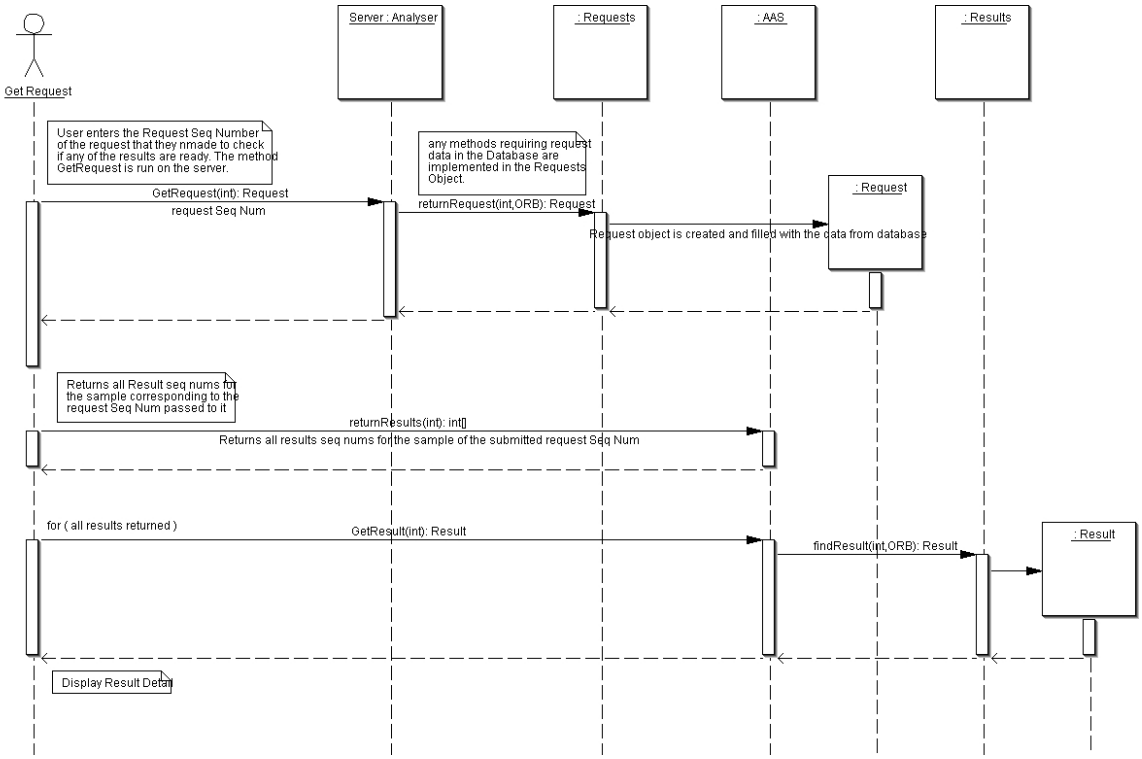

Figure 42: Sequence diagram for retrieving requested results

Appendix D: Blood Test Reference Ranges

The following table is a list of reference ranges for various blood tests [70]. The Glucose reference range was used in the implementations in this thesis.

Test	Reference Range (conventional units)
Acidity (pH)	7.35 - 7.45
Alcohol	0 mg/dL (more than 0.1 mg/dL normally indicates intoxication) (ethanol)
Ammonia	15 - 50 µg of nitrogen/dL
Amylase	53 - 123 units/L
Ascorbic Acid	0.4 - 1.5 mg/dL
Bicarbonate	18 - 23 mEq/L (carbon dioxide content)
Bilirubin	Direct*:* up to 0.4 mg/dL Total*:* up to 1.0 mg/dL
Blood Volume	8.5 - 9.1% of total body weight
Calcium	8.5 - 10.5 mg/dL (normally slightly higher in children)
Carbon Dioxide Pressure	35 - 45 mm Hg
Carbon Monoxide	Less than 5% of total hemoglobin
CD4 Cell Count	500 - 1500 cells/µL
Ceruloplasmin	15 - 60 mg/dL
Chloride	98 - 106 mEq/L
Complete Blood Cell Count (CBC)	Tests include: hemoglobin, hematocrit, mean corpuscular hemoglobin, mean corpuscular hemoglobin concentration, mean corpuscular volume, platelet count, white Blood cell count.
Copper	Total*:* 70 - 150 µg/dL
Creatine Kinase (CK or CPK)	Male*:* 38 - 174 units/L Female*:* 96 - 140 units/L
Creatine Kinase Isoenzymes	5% MB or less
Creatinine	0.6 - 1.2 mg/dL
Electrolytes	Test includes: calcium, chloride, magnesium, potassium, sodium.
Erythrocyte Sedimentation Rate (ESR or Sed-Rate)	Male*:* 1 - 13 mm/hr Female*:* 1 - 20 mm/hr
Glucose	Tested after fasting: 70 - 110 mg/dL

Hematocrit	Male*:* 45 - 62% Female*:* 37 - 48%	
Hemoglobin	Male*:* 13 - 18 gm/dL Female*:* 12 - 16 gm/dL	
Iron	60 - 160 µg/dL (normally higher in males)	
Iron-binding Capacity	250 - 460 µg/dL	
Lactate (lactic acid)	Venous*:* 4.5 - 19.8 mg/dL Arterial*:* 4.5 - 14.4 mg/dL	
Lactic Dehydrogenase	50 - 150 units/L	
Lead	40 µg/dL or less (normally much lower in children)	
Lipase	10 - 150 units/L	
Zinc B-Zn	70 - 102 µmol/L	
Lipids:		
Cholesterol	Less than 225 mg/dL (for age 40-49 yr; increases with age)	
Triglycerides	10 - 29 years	53 - 104 mg/dL
	30 - 39 years	55 - 115 mg/dL
	40 - 49 years	66 - 139 mg/dL
	50 - 59 years	75 - 163 mg/dL
	60 - 69 years	78 - 158 mg/dL
	> 70 years	83 - 141 mg/dL
Liver Function Tests	Tests include bilirubin (total), phosphatase (alkaline), protein (total and albumin), transaminases (alanine and aspartate), prothrombin (PTT)	
Magnesium	1.5 - 2.0 mEq/L	
Mean Corpuscular Hemoglobin (MCH)	27 - 32 pg/cell	
Mean Corpuscular Hemoglobin Concentration (MCHC)	32 - 36% hemoglobin/cell	
Mean Corpuscular Volume (MCV)	76 - 100 cu µm	
Osmolality	280 - 296 mOsm/kg water	
Oxygen Pressure	83 - 100 mm Hg	

Oxygen Saturation (arterial)	96 - 100%	
Phosphatase, Prostatic	0 - 3 units/dL (Bodansky units) (acid)	
Phosphatase	50 - 160 units/L (normally higher in infants and adolescents) (alkaline)	
Phosphorus	3.0 - 4.5 mg/dL (inorganic)	
Platelet Count	150,000 - 350,000/mL	
Potassium	3.5 - 5.0 mEq/L	
Prostate-Specific Antigen (PSA)	0 - 4 ng/mL (likely higher with age)	
Proteins:		
Total	6.0 - 8.4 gm/dL	
Albumin	3.5 - 5.0 gm/dL	
Globulin	2.3 - 3.5 gm/dL	
Prothrombin (PTT)	25 - 41 sec	
Pyruvic Acid	0.3 - 0.9 mg/dL	
Red Blood Cell Count (RBC)	4.2 - 6.9 million/µL/cu mm	
Sodium	135 - 145 mEq/L	
Thyroid-Stimulating Hormone (TSH)	0.5 - 6.0 µ units/mL	
Transaminase:		
Alanine (ALT)	1 - 21 units/L	
Aspartate (AST)	7 - 27 units/L	
Urea Nitrogen (BUN)	7 - 18 mg/dL	
BUN/Creatinine Ratio	5 - 35	
Uric Acid	Male	2.1 to 8.5 mg/dL (likely higher with age)

	Female	2.0 to 7.0 mg/dL (likely higher with age)
Vitamin A	30 - 65 µg/dL	
White Blood Cell Count (WBC)	4,300 - 10,800 cells/µL/cu mm	

Table 12: Blood test reference ranges

Appendix E: Publications

Owen Lynch, John McGrory and Eugene Coyle, *Design of Mobile Phone Applications for Point of Care Test Result Validation*, IASTED International Conference on Telehealth, Banff, Canada, 19 – 21 July 2005.

DESIGN OF MOBILE PHONE APPLICATIONS FOR POINT OF CARE TEST RESULT VALIDATION

Owen Lynch, John McGrory and Eugene Coyle
School of Control Systems and Electrical Engineering
Dublin Institute of Technology
Kevin St.
Dublin 8
Ireland
{owen.lynch, john.mcgrory, eugene.coyle}@dit.ie

ABSTRACT
Patients with many different conditions are required to take the management of their condition into their own hands and perform Point of Care Testing (POCT) at home. However, this raises quality control issues that would not arise in a clinical setting, since the sample acquisition and testing procedures are not overseen by professional hospital staff. Another major issue, the main focus of this research, is that results from such tests are not clinically validated to ensure that they are plausible for that patient at the time of testing. In hospital, tests taken by clinicians are validated by hi-tech computerised validation systems, before a diagnosis is made. Patients at home must often use the results of tests they take to determine medication dosage or monitor their condition, but these results do not undergo a validation procedure. Thus, there is a need to implement a system of result validation, either locally or by the hospital validation system itself, for people testing at home with POCT devices. This paper describes how mobile phone applications may be used to link patients, who manage their condition in the home, with the hospital information system (HIS) to upload and validate their results.

KEY WORDS
Point of Care, Home Health Care, J2ME, Mobile Application, Clinical Validation, Data Transmission

1. Introduction

1.1 Point of Care Testing

Point of Care testing in the home is a rapidly growing area in the healthcare arena. It gives patients an opportunity to manage their own conditions and can reduce their length of stay in hospital. There is also a cost reduction associated with the release of patients to their home for continuance of their healthcare. With a rapidly ageing worldwide population, and the older share of the population set to double by 2030 [1], there is a need to increase POCT at home in the health care system. There are a variety of ailments of the aged for which doctors can utilise clinical laboratory tests, and providing POCT to homebound patients could have a significant impact on their condition [2]. The variety of test apparatus for patients who manage their conditions at home is as diverse as the ailments themselves, ranging from simple urine dipsticks and blood pressure arm cuffs to complicated ECG devices. Many patients also use blood-testing units, such as glucometers for diabetics taking insulin to check glucose levels or blood coagulation (INR) meters for people taking anticoagulant medication, such as warfarin. The management of diabetes in the USA costs $100 billion annually and has many secondary disorders associated with it. However, management of the condition with POCT would prevent many of these disorders [2]. It is not only chronically ill patients and the ageing who use POCT, women during pregnancy and patients recently released from hospital may also have to monitor biological signs.

1.2 Clinical Testing and Result Validation

In hospitals and other similar clinical settings, tests are often required to be performed on specimen samples, such as blood or urine, which are taken from patients. The results of these tests provide invaluable information to doctors and clinicians for assessing a condition or making a diagnosis. The samples are taken from the patient by trained staff and in a hygienic manner that is well established, labelled and sent to the lab where the requested tests are performed by high precision lab testing equipment. The clinical laboratory applies gold standard testing regimes on each sample to obtain a result. The gold standard is considered the most specific test for a given sample and is completed according to a strict workflow list. Test results are then validated by a computerised validation system to determine their plausibility. This validation system may be bespoke software rules written by the hospital's own professionals or a commercially available system such as LabRespond [3]. Clinical validation systems play an important role in large automated laboratories and allow professionals to concentrate on problematic cases. These systems typically use a rule base to validate the result data as well as

clinical information such as past result values, sex and age to determine the believability of the result [4]. If there is a problem with the result and it is deemed invalid, it is marked for closer inspection by a laboratory professional. Further action can then be taken or the test can be re-ordered.

1.3 Home Testing Issues

For patients who perform tests at home with POCT equipment, there are some major issues, which this research addresses. First, the testing quality achieved by patients operating self-monitoring instruments is less when compared to a technician using the same equipment [5]. Second, there is no independent validation of the test results as described above. Thus decisions made as a result of a home test may be the wrong ones, which could potentially lead to complications for the patient. The consequence of producing the wrong results for INR could result in haemorrhaging or death of a warfarin patient [2]. Finally, the hospital professionals and doctors are unaware of their patients' progress between hospital visits. Patients may be scheduled to report to the hospital for weekly or monthly check-ups. However if a complication arises between visits, it may go unnoticed. Another issue with home based POCT is that prescribed drugs may interact with and affect test results. If the hospital is aware of this, it can allow for it in the validation algorithm and make changes to its outcome if necessary.

1.4 Proposed Solution

A major challenge of POCT is the integration of test results into hospital information systems (HIS) and electronic patient records (EPR). The need for connectivity between POCT devices and HIS is well recognised and benefits have been documented [6]. This research endeavours to bridge the gap between the patient at home and the hospital responsible for the treatment of the patient. The technologies considered will be discussed and then some solutions proposed for connecting patients with the hospital system for validation of results and the population of data to their EPR, using a mobile phone.

2. Mobile Phone and Java Technologies

2.1 Introduction

With the worldwide mobile phone subscriber numbers passing the 1.5 billion mark in 2004 and predicted to hit the 2 billion mark as early as July 2006 [7], it is clear that mobile phones are now a familiar tool to nearly everyone and are a necessary tool in everyday life for some. Phones are now more than just simple voice communicators and are evolving into sophisticated mini-computers, capable of running small to medium sized applications. Research has been carried out to investigate different mobile technologies that could be used as part of the solution for validating results of patients using POCT equipment.

2.2 Relevant Mobile Phone Technologies

The available mobile technologies were investigated and compared to one another to see which would suit this problem the most. Three of the technologies considered for use as a possible solution were Short Message Service (SMS), Wireless Application Protocol (WAP) and Java 2 Micro Edition (J2ME).

SMS was first considered but was soon ruled out as a realistic solution. SMS is mostly associated with short text-based messages for simple communication between phone subscribers, but a stream of SMS messages can be used to transmit any digital data. However, SMS is a store-and-forward service and there is no guarantee that the information will be delivered in a timely manner [8]. As POCT is time critical, this would not be acceptable.

WAP is an open specification that lets wireless devices easily interact with services and allows users to access the Internet [9]. So it could be used, for example, by patients to access a hospital server, submit test results and have relevant information returned to them, both textually and graphically. It uses the GPRS network, which is an always-on service and provides the highest possible transmission rates in GSM networks. However, there are also big limitations in using WAP as a solution. It requires a constant connection to the network and thus an off-line solution for validation would not be possible.

2.3 Java 2 Micro Edition (J2ME)

J2ME is a lighter version of the standard edition of Java designed specifically for developing applications on wireless communications devices with limited memory sizes, such as standard mobile telephones. It is a rapidly developing technology; there are literally hundreds of phones capable of running J2ME [10] and it is becoming a standard feature on mobile phones. J2ME applications are developed as MIDlets, MIDlets are to wireless what Java Applets have been to the web. J2ME applications can run on the phone without a network connection and are capable of making secure HTTP (HTTPS) connections to Internet servers and parsing the response. They are capable of storing data persistently, displaying data graphically and can perform relatively complex computation. Another beneficial aspect to J2ME is its ability to handle and parse XML, a format for structured documents and data that facilitates the interchange of data between computer applications. Because of these and its many other features J2ME has been chosen to develop the applications for ensuring high quality result validation as will be discussed in the next section.

2.4 Java Servlets

As noted above, J2ME applications have the ability to make HTTP connections and can handle and parse XML documents. This means that they can wrap any data into an XML document and send it to a server. Java Servlet technology works using this technique. It provides a mechanism for extending the functionality of a web server and accessing business systems [11]. Java Servlets have access to all of the existing Java application programming

interfaces (APIs), so that powerful server programs can be accessed by simple client applications remotely. Java Servlet technology has been chosen to implement the test hospital server programs and will be used for validating the results, providing software updates for the client application and storing the patient results in the EPR.

3. System Design

3.1 Introduction

Three versions of the application have been considered as possible solutions to the problem. Each version of the system consists of a J2ME client application, which runs on the phone, and a server application, which will be implemented as a Java Servlet. In the first version, the application will perform the validation of the result on the phone, only connecting to the hospital to update the validation rules and algorithm if necessary and to upload the result to the hospital database. In the second version the hospital server will do all of the validation computation, making the mobile client application much smaller. The third version splits the workload between the phone and the server, so some initial computation may be done on the phone but the application can remotely call procedures on the server for more heavy computation and uploading of result data. All three versions will include some pre-test instructions or procedures to help ensure a more accurate and reliable test result is obtained.

3.2 Version 1: All Computation on the Phone

In this case the client application is equipped with the rule base and algorithm to validate the patient's result locally. The rules could be tailored specifically for a patient and their condition if necessary. This method is suitable for result types where the validation procedure is relatively simple. Once the validation is finished, the application connects to the hospital and transmits the result data in XML format for storage on the hospital database. In the case of an invalid result, it will be flagged so a clinician can view it and contact the patient with further advice. A simple workflow of the system is shown in Figure 1. If previous result values are needed to validate a result they will be stored locally on the phone but can be retrieved from the hospital database if necessary. The validation algorithm may need to be updated or changed by the hospital technicians from time to time, so the application is designed with this in mind and has functionality to check for an update. If one is available, it downloads the new data and uses it for validation of results. Although all three versions of the application will need to make network connections, this one will generally use the least "air-time" as it does not need to wait for a response from the hospital validation system to validate the results. However, due to the limited processing power on a phone it is only suitable for the more simple validation methods.

3.3 Version 2: All Computation on the Server

This version of the application means there is less work done by the client application on the actual validation of the result. Information entered by the user on the phone is wrapped in an XML document and sent to the server over the HTTP connection, the result is then validated on the server by the validation process. If the result is valid it is stored on the patients EPR. If invalid, it is marked for inspection by the relevant hospital staff. A response for the client is generated and the outcome of the validation is included in it. Figure 2 shows a simplified workflow of how the client and servers would work together in the system. As results are validated by the most up to date validation service on the hospital server, there is less need to continually check the client application for updates in this version.

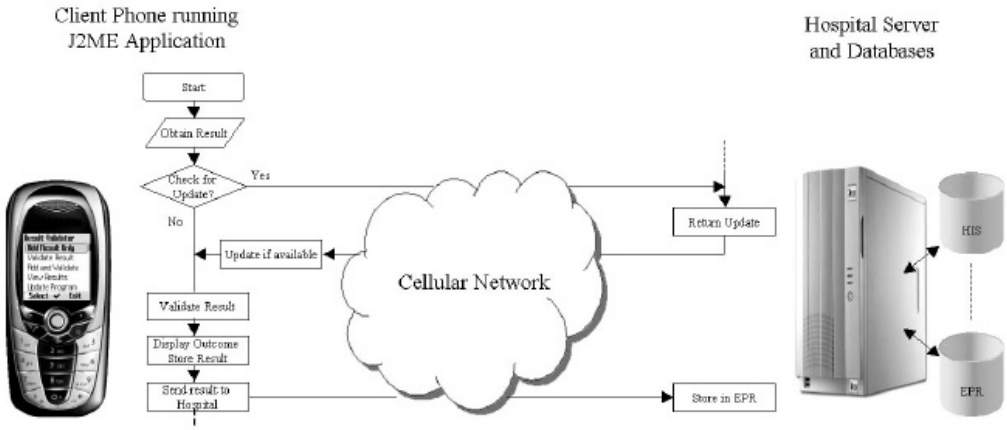

Figure 1: Simplified workflow of version one.

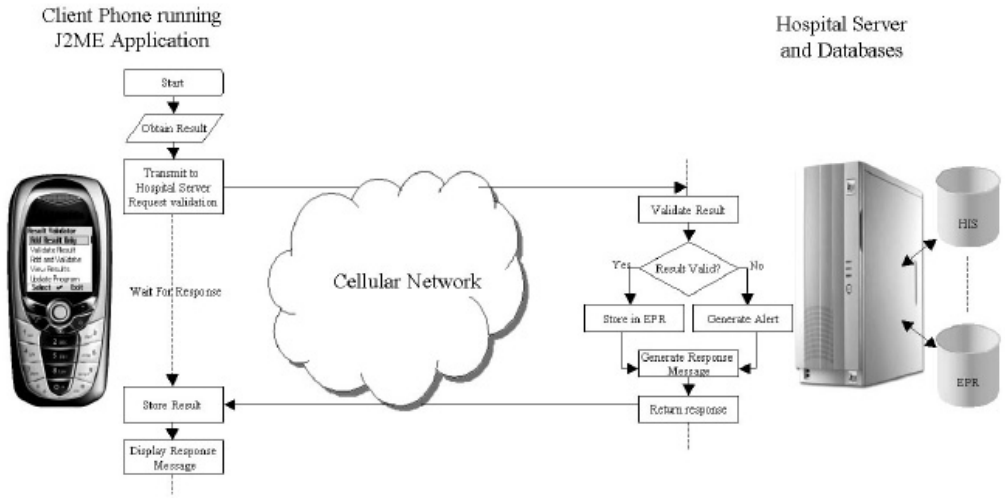

Figure 2: Simplified workflow of version two.

3.4 Version 3: Shared Computation

This version is potentially the most sensible one. It is similar to the second one but is a solution that utilises distributed computing technologies. When the result to be validated is entered this application will invoke the method for validation on the server remotely. To the user it will seem like the application is doing the computation locally but in fact it will be done by the server application. This distributed solution means the client application can be smaller than the first version, which is more suitable for a J2ME device. It also means, as in the previous version, that the result is being validated by the hospital validation service, which will always be the most up to date version. XML-RPC is a lightweight XML-based protocol for remote method invocation over HTTP and is being considered for use in this version of the system.

4. Implementation

4.1 Introduction

The basic designs for the three versions of the system have now been outlined. Proof of concept implementations for both the client and server side applications are being carried out. When finished, the three implantations will be tested and benchmarked against one another so that the strengths and weaknesses of each can be identified. With this information, the optimum version may be identified and improvements can be made. All three versions are designed with simplicity of use in mind so that the system will be manageable by elderly patients and people who are not technically educated.

4.2 Build and Test of The System

For the proof of concept, the J2ME applications are tested using various mobile phone emulators as well as the J2ME Wireless Development Toolkit. The Servlets written are running on an Apache Tomcat server. The Database on the server side has been built using the MySQL RDMS. The result data to be sent from client to server is wrapped to XML format and based on the HL7 format for laboratory results.

The selection of the optimal design from the three versions described will be based on the results of comprehensive testing and benchmarking of the systems. There are good reasons for implementing each version but testing each one and comparing the results of all three will confirm which ones are viable solutions. Factors being considered to determine the optimum system will be the accuracy of the result validation compared to the gold standard, the time it takes to receive a validation after the result is entered, and the efficiency in terms of network traffic.

4.3 Application Deployment

For all three cases the method for deploying the application will be the same. J2ME applications can be deployed to mobile phones over the air (OTA) and thus users are not required to have data cables for their handsets. Once downloaded, the application can be run on the phone without connecting to the network. The process will work as follows. The patient downloads the application to their phone from the hospital server. The application runs locally but can make HTTP connection to the hospital to upload data or validate the result. The client application receives data back from the hospital and

can perform additional computation if necessary and store the result locally.

5. Conclusion

With the increase in the demand for POCT in the home, there is a great opportunity for work in the area of patient-hospital communication. There is already much work being done on the communication of data between POCT units and wireless devices such as mobile phones and PDA's. The applications outlined in this paper take this communication to the next level and allow the results to be transmitted to the hospital. The three versions described in this paper give good guidelines as to how such a system may be realised The implementation of these applications will confirm the benefits. The wireless industry is rapidly growing and with the roll out of 3G networks already in operation there is great scope for expanding these applications, thus ensuring greater patient care and less wasted time.

References:

[1] K. Kinsella & D. R. Phillips, Global Ageing: The Challenge of Success, *Population Bulletin, 60*(1), 2005.
[2] C. A. Lehmann, The Future of Home Testing – implications for traditional laboratories, *Clinica Chimica Acta, 323*(1-2), 2002, 21-36.
[3] W. P. Oosterhuis, J. L. M. Herman & H. M. J. Goldschmidt, Evaluation of LabRespond, a new Automated Validation System for Clinical Laboratory Test Results, *Clinical Chemistry 46*(11), 2000.
[4] G. Boran, P. Given & R. O'Moore, Patient Result Validation Services, *Computer Methods and Programs in Biomedicine, 50*(2), 1996, 161-168.
[5] S. Skeie, G. Thue, K. Nerhus & S. Sandberg, Instruments for Self Monitoring of Blood Glucose: Comparisons of Testing Quality Achieved by Patients and a Technician, *Clinical Chemistry, 48*(7), 2002, 994-1003.
[6] M. Taylor, J.H. Nichols & J. Saltz, POCT Connectivity, Opening the door to a laboratory without walls, *American Clinical Laboratory*, July 2000.
[7] S. Josifovska, Where Next For The Handset?, *IEE Review, 50*(12), 2004, 36-39.
[8] T. Knyziak, W. Winiecki, The new prospects of distributed measurement systems using Java 2 Micro Edition mobile phone, *Computer Standards and Interfaces*, Article in press, available online at www.sciencedirect.com, 2005.
[9] K. Read & F. Maurer, Developing Mobile Wireless Applications, *IEEE Internet Computing 7*(1), 2003, 81-86.
[10] Java 2 Platform, Micro Edition (J2ME), http://java.sun.com/j2me/
[11] Java Servlet Technology, http://java.sun.com/products/servlet/

John McGrory, Owen Lynch and Eugene Coyle, *Design of a Wireless System for Patient-Hospital Communication and Result Validation in Point of Care Testing*, International Conference of Computational Intelligence and Multimedia Applications ICCIMA'05, Las Vegas, USA, 16 – 18 August 2005.

Design of a Wireless System for Patient-Hospital Communication and Result Validation in Point of Care Testing

John McGrory, Owen Lynch and Eugene Coyle
Dublin Institute of Technology, Ireland
john.mcgrory@dit.ie, owen.lynch@dit.ie and Eugene.coyle@dit.ie
http://www.dit.ie

Abstract

This paper discuses mobile phone (cell phone) and wireless applications for linking patients who manage their healthcare outside the hospital using Point of Care Testing (POCT) to hospital information systems (HIS). Certain medical conditions require patients to manage their healthcare by performing on themselves POC testing and act faithfully on the result. This raises quality control issue, as these POC samples and testing procedures are not independently overseen by professional hospital staff. In hospitals, samples taken by clinicians are validated by hi-tech computerised validation systems to ensure plausibility, before physicians rely on them. Patients in the home must often use results from these POCT to determine medication dosage or to monitor their condition. Thus, there is a need to implement a system of result validation, either locally or by the hospital validation system itself, for people testing with POCT devices.

1. Introduction

1.1 Point of care testing

POCT gives patients an opportunity to manage their own conditions, reduce hospital stays and minimise related costs associated with hospital testing for example, patient time and travel. With an ageing worldwide population, set to double by 2025 [1], there is a need to increase the use of POCT in the healthcare system model. POCT to homebound patients could have significant impact on their condition [2]. The diversity of POCT apparatus available is as varied as the ailments themselves ranging from simple urine dipsticks and blood pressure arm cuffs to complicated ECG devices. Patients also use blood-testing units, such as glucometers for diabetics to check glucose levels or blood coagulation (INR) meters for people taking anticoagulant medication, such as warfarin. The management of diabetes in the USA costs $100 billion annually and has many secondary disorders associated with it, however management of the condition with POCT would minimise many of these disorders [2].

1.2 Clinical testing and result validation

For a physician to manage an illness it is common practice to gather patient samples for analysis (blood, urine, faeces etc.). The clinical laboratory applies gold standard testing regimes on each sample to obtain a result. The gold standard is considered the most specific and sensitive test for this sample type and completed strictly according to a workflow list. Once a result is obtained the laboratory begins checking if this result is plausible for this patient in this instance. Result validation is generally divided into two areas, namely "Technical validation" and "Clinical validation".

Technical validation checks for instrument calibration, sample tolerance, repeatability, reliability and certification. This insures the results are from a calibrated, certified process, the results are consistent and samples are not contaminated. Clinical validation on the other hand is when the result is checked for plausibility. They ensure results are relevant to the patient and the complaint being investigated.

Table 1 illustrates general tests undertaken by a clinical laboratory in the validation of a sample, specifically blood.

Type of validation checks	Description
Calibration	Instrument checked is calibrated and working correctly.
Instrument specific checks	Vary depending on instrument and analytical process used to generate the result. Often carried out either by the instrument itself or manually by the instrument operator.
Internal consistency checks	The consistency between pathophysiologically related variables is examined
Basic validation checks	Data checked against age and sex related reference ranges, pathological limits as set down by the laboratory based on international standards.
Delta checks	Previous results are compared with the current results using various techniques.
Sample mix-ups	When one sample is given the identity of another, usually an adjacent sample on the workbench

Table 1, Advanced Validation Checks [4] Modified

When the checks are completed and if the sample results are still plausible the data is considered validated and populated into the patient's healthcare record. If at any stage the results are not plausible then the patient information is forwarded to the laboratory manager were a retest of the sample can be authorized or a new sample requested. POCT should be validated in a similar fashion where possible to the clinical laboratory test for consistency.

1.3 Home testing issues

For patients who perform tests at home with POCT equipment, there are four major issues which this research addresses.

(i) Testing quality achieved by patients operating self monitoring instruments is lesser when compared to a technician using the same instrument [11].

(ii) No independent validation of the test results. Thus decisions made as a consequence of POCT may be erroneous, potentially leading to patient complications or death.

(iii) The hospital professionals and doctors are unaware of patient progress between hospital visits. Complications arises between visits, it may go unnoticed.

(iv) Prescribed drugs and diet interacting and affecting test results. For example, large amounts of broccoli, spinach, and other green leafy vegetables high in vitamin K promote the formation of blood clots counteract the effects of warfarin, and other drugs given to prevent clotting.

Hospital validation servers made aware of these can change validation parameters or validation algorithm, tuning its accuracy and making comments useful to the patient.

1.4 Proposed solution

A major challenge of POCT is the integration of test results into hospital information systems (HIS) and electronic patient records (EPR). The need for connectivity between POCT devices and HIS is well recognised and benefits have been documented [5]. This research endeavours to bridge the gap between the patient at home and the hospital responsible for the treatment of the patient. Some solutions are proposed for connecting patients with the hospital system for validation of results and the population of data to their EPR, using a mobile phone.

2. Mobile phone and java technologies

2.1 Introduction

With the worldwide mobile phone subscribers passing 1.5 billion in 2004 and predicted to reach 2 billion by July 2006 [6], it is clear mobile phones are a familiar tool and are becoming a necessary tool in everyday life for many people. Phones are now more than just simple voice communicators and are evolving into sophisticated mini-computers, capable of running small to medium sized applications. Research has been carried out to investigate different mobile technologies that could be used in the solution for validating results of patients using POCT equipment.

2.2 Relevant mobile phone technologies

Three of the technologies considered for use as a possible solution were Short Message Service (SMS), Wireless Application Protocol (WAP) and Java 2 Micro Edition (J2ME).

SMS is a short text-based message for simple communication between phone subscribers, but a stream of SMS messages can be used to transmit any digital data. However, SMS is a store-and-forward service and there is no guarantee that the information will be delivered in a timely manner [7]. As POCT is time critical, this would not be acceptable.

WAP is an open specification allowing wireless devices easily interact with services and permits users to access the Internet [8]. For example, patients accessing the hospital server submit test results and have relevant information returned to them, both textually and graphically. It uses the GPRS network, which is an "always-on" service and provides the highest possible transmission rates in GSM networks. However, there are also limitations in using WAP as a solution. It requires a constant connection to the network and thus an off-line solution for validation would not be possible.

J2ME is lighter than standard Java designed specifically for developing applications on wireless communications devices with limited memory sizes, such as mobile telephones. There are hundreds of phones capable of running J2ME [9] and it is becoming a standard feature on mobile phones. J2ME applications are developed as MIDlets, MIDlets are to wireless what Java Applets have been to the web. J2ME applications can run on the phone without a network connection and are capable of making HTTP connections to Internet servers and parsing the response. They are capable of storing data persistently, displaying data graphically and can perform relatively complex computation. J2ME has the ability to handle and parse XML, a format for structured documents and data that facilitates the interchange of data between computer applications. Explorative work on simple agents has allowed the automated selection of the most suitable hospital service server available from a range of services for the sample result to be validated using XML. Therefore, J2ME has been chosen to develop the applications for ensuring high quality result validation.

2.3 Java servlets

J2ME applications can wrap any data into an XML document and send it to a server. Java Servlet technology operates using this technique. It provides a mechanism for extending the functionality of a web server and accessing business systems [10]. Java servlets have access to all of the existing Java application programming interfaces (APIs), so that powerful server programs can be accessed by simple client applications remotely. Java Servlets have been chosen to implement the test hospital server programs and will be used for validating the results [12], providing software updates for the client application and storing the patient results in the EPR.

3. System design

3.1 Introduction

Three versions of the application have been considered. All applications consist of a J2ME client application, running on the phone, and server application implemented as a Java Servlet. For simplicity we have omitted the pre-test instruction/procedure sets which are necessary irrespective of version chosen. In the first version, the phone application performs the validation of the result, only connecting to the hospital to update the validation procedure, and upload the result to the hospital database. In the second version the hospital server processes all validation computation, making the mobile client application much smaller. The third version divides the workload between the phone and server, so some initial computation may be done on the phone, but the application can remotely call procedures on the server for more heavy computation and uploading of result data.

3.2 Version 1: All computation on the phone

The client application runs the rule base and algorithm to validate the patient's result locally. The rules could be tailored specifically for a patient. This method is suitable for result types where the validation procedure is relatively simple. PDA and mobile phone processor speeds vary between 200-550MHz, with cache memory and bus speeds characteristics clearly restricting their processing capability. In addition programmers have no homogeneous access to lower level apparatus, for example camera or battery power remaining features on the phone without resorting to vendor/phone specific java class sets. This restricts the development of the "all on phone" processing capability. Once the validation is completed, the application connects to the hospital and transmits the result data in HL7 XML format [12] for storage on the hospital database. A simple workflow of the system is shown in Figure 2. Previous results will be stored locally on the phone, but can be retrieved from the hospital database also. The validation algorithm may be updated and changed by the hospital technicians at any time; the application designed allows for this with functionality to check for an update. If one is available, it downloads the new data and uses it for validation of results. Although all three versions of the application will need to make network connections, this one will generally use the least "air-time" as it does not need to wait for a response from the hospital validation system to validate the results.

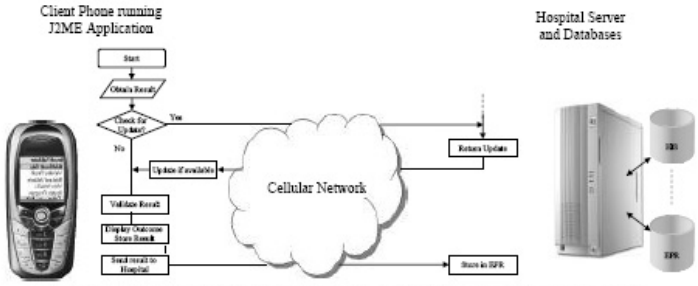

Figure 1: Simplified workflow for the first version of the system.

3.3 Version 2: All computation on the server

This application minimises the work done by the client application on the actual validation of the result. Information entered by the user on the phone is wrapped in an XML document and sent to the server over the HTTP connection. The result is then validated on the server by the validation process. If the result is valid it is stored on the patients EPR. If invalid, it is marked for inspection by the relevant hospital staff. A response for the client is generated and

the outcome of the validation is included in it. Figure 2 depicts a simplified workflow of how the client and servers would work together in the system. This client application needs to connect the patient's result with the hospital server ensuring results are validated by the same quality validation service as the one validating results of hospital tests.

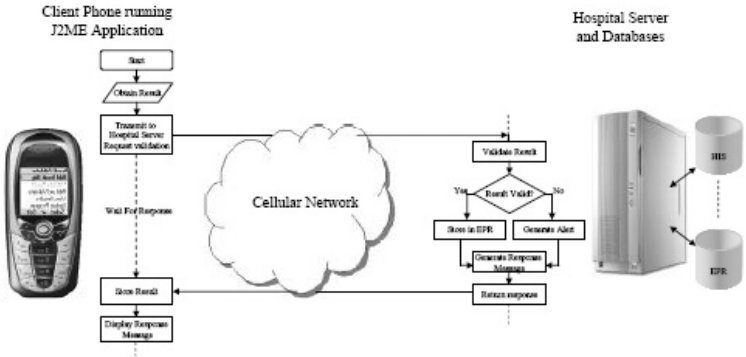

Figure 2: Simplified workflow for the second version of the system.

3.4 Version 3: Shared computation

This version could potentially be the most useful/appropriate application and is similar to the second version described above. When results to be validated enter this application they invoke the method for validation on the server remotely. This could be achieved using mobile agents. The agent architecture brokers communications with the most suitable hospital validation service. To the user it will seem like the application is doing the computation locally but in fact it will be done by the server application. This distributed solution means the client application can be smaller than the first version, which is more suitable for a J2ME device. It also means, as in the previous version, the result is being validated by the hospital validation service, which will always be the most up to date version. XML-RPC is a lightweight XML-based protocol for remote method invocation over HTTP and is being considered for use in this version. Figure 2 shows a simplified workflow for this third version.

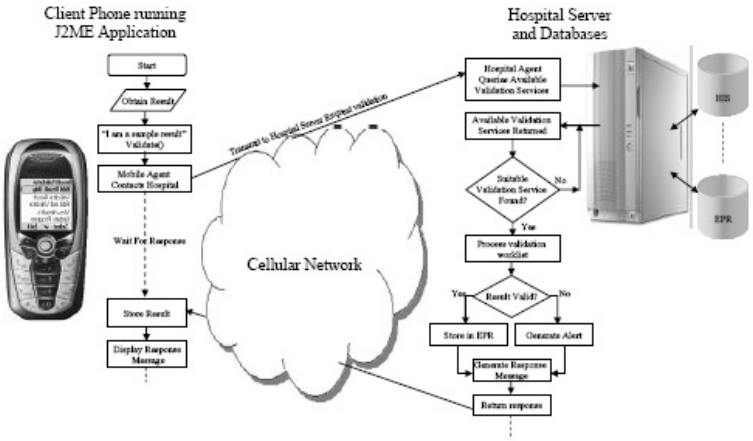

Figure 3: Simplified workflow for the third version.

4. Planned implementation

4.1 Building and testing the system

The basic designs for the three versions of the system have now been outlined. The implementations for both the client and server applications are being carried out and the selection of the optimal design from the three versions described will be based on the results of comprehensive testing and benchmarking of the systems. There are good reasons for implementing each version by testing each one and comparing the results of all three will confirm which ones are viable solutions. Factors being considered to determine the optimum system will be; the *accuracy* of the result validation compared to the gold standard, *processing time*, *scalability* and *network traffic efficiency*.

4.2 Application deployment

For all three cases the method for deploying the application will be the same. J2ME applications can be deployed to mobile phones over the air (OTA) and thus users are not required to have data cables for their handsets. The process will work as follows; the patient downloads the application to their phone from the hospital server. The application runs locally but can make HTTP connection to the hospital to upload data or validate the result. The client application receives data back and can perform additional computation if necessary and store the result locally.

5. Conclusion

It has been shown that with the increase in the demand for POCT in the home there is a great opportunity for work in the area of patient-hospital communication. The applications outlined in this paper give a guideline as to how such a system may be realised. With the roll out of 3G networks already in operation there is great scope for expanding these applications, thus ensuring greater patient care and less time wasted. With the increased volume of information now being gained from POCT there is an increased demand on *when, where* and *how* the information is obtained and stored. Agent brokering theory is being used to improve the selection and execution difficulties experienced with distributed technologies [13].

6. References

[1] World Health Organization, http://www.who.int/hpr/ageing/ageing.pdf accessed March 2005.

[2] Lehmann, C. A., "The Future of Home Testing – implications for traditional laboratories", Clinica Chimica Acta, Vol 323 pages 21 - 36, September 2002.

[3] Oosterhuis W. P., Herman J. L. M., Goldschmidt H. M. J., "Evaluation of LabRespond, a new Automated Validation System for Clinical Laboratory Test Results", Clinical Chemistry Volume 46, Issue 11, November 2000.

[4] Boran G., Given P., O'Moore R., "Patient Result Validation Services, Computer Methods and Programs in Biomedicine", Vol 50 Page 161 – 168, 1996.

[5] Taylor, M., Nichols, J.H., Saltz, J., "POCT Connectivity, Opening the door to a laboratory without walls", American Clinical Laboratory, July 2000.

[6] Josifovska S., "Where Next For The Handset?", IEE Review Volume 50, Number 12, December 2004.

[7] Knyziak T., Winiecki W., "The new prospects of distributed measurement systems using Java 2 Micro Edition mobile phone", Computer Standards and Interfaces, Article in press, available online February 2005 at www.sciencedirect.com

[8] Read K., Maurer F., "Developing Mobile Wireless Applications", IEEE Internet Computing, January/February 2003, http://computer.org/internet/

[9] Java 2 Platform, Micro Edition (J2ME), http://java.sun.com/j2me/

[10] Java Servlet Technology, http://java.sun.com/products/servlet/

[11] Skeie Svein, Geir Thue, Kari Nerhus and Sverre Sandberg "Instruments for Self-Monitoring of Blood Glucose: Comparisons of Testing Quality Achieved by Patients and a Technician" published by Clin. Chem., Jul 2002; 48: 994 - 1003.

[12] HL7 messaging system schema, http://www.hl7.org, accessed Dec 04

[13] Agent principles, http://agent.omg.org, accessed Dec 04.

References

[1] K. Kinsella and D. R. Phillips, "Global Ageing: The Challenge of Success," *Population Bulletin*, vol. 60 (1), 2005.

[2] O. Lynch, J. McGrory, and E. Coyle, "Design of Mobile Phone Applications for Point of Care Test Result Validation," presented at IASTED International Conference on Telehealth, Banff, Canada, 19 - 21 July 2005.

[3] A. MacFarlane, A. W. Murphy, and P. Clerkin, "Telemedicine services in the Republic of Ireland: An evolving policy context," *Health Policy*, vol. Elsevier (www.sciencedirect.com), 2005.

[4] J. M. Fitzmaurice, "Telehealth Research and Evaluation: Implications for Decision Makers," presented at Pacific Medical Technology Symposium, 17-20 Aug 1998.

[5] S. Senapati and A. P. Advincula, "Telemedicine and robotics: Paving the way to the globalization of surgery," *International Journal of Gynecology & Obstetrics*, vol. 91 (3), 2005.

[6] E. Coiera, *Guide To Health Informatics*, 2nd ed: Arnold Publishers, 2003.

[7] B. Brown, A Brief History of Telemedicine, *Telemedicine Information Exchange*, http://tie.telemed.org/articles.asp, 1995, Accessed September 2005

[8] W. R. Hersh, M. Helfand, J. Wallace, D. Kraemer, P. Patterson, S. Shapiro, and M. Greenlick, "Clinical outcomes resulting from telemedicine interventions: a systematic review," *BMC Medical Informatics and Decision Making*, vol. 1 (5), 2001.

[9] J. Ansell, A. Jacobson, J. Levy, H. Voller, and J. M. Hasenkam, "Guidelines for implementation of patient self-testing and patient self-management of oral anticoagulation. International consensus guidelines prepared by International Self-Monitoring Association for Oral Anticoagulation," *International Journal of Cardiology*, vol. 99 pp. 37 - 45, 2005.

[10] "Mobile phones may supply link to better health," in *The Irish Times*. Dublin, 7th October 2005.

[11] M. Kramer, T. Norgall, and T. Penzel, *Short Strategic Study: Strategies for harmonisation and integration of device-level and enterprise-wide methodologies for communication as applied to HL7, LOINC and ENV 13734*: CEN/TC 251, 2001.

[12] A. Cronin, "A Wireless Data Logger," MPhil, Faculty of Engineering, Dublin Institute of Technology, 2005

[13] J. H. Van Bemmel and M. A. Musen, *Handbook of Medical Informatics*, 1st ed: Springer-Verlag, 2000.

[14] *ASTM International*, http://www.astm.org/, Accessed November 2005

[15] *ENV 13728: Health informatics - Instrument interfaces to laboratory information systems*: CEN/TC 251, 1999.

[16] F. Knox, "The Design and Implementation of an Integrated Networked Clinical Analyser," M.Sc., Staffordshire University,

[17] P. Wilkinson, "2010 vision," *NHS Magazine*, 2004.

[18] S. Rogerson, "Electronic Patient Records, ETHIcol," *IMIS Journal*, vol. 10 (5), 2000.

[19] What is HL7?, *Health Level Seven (HL7) - official website*, http://www.hl7.org/, Accessed October 2005

[20] E.-W. Huang, D.-W. Wang, and D.-M. Liou, "Development of a Deterministic XML Schema by Resolving Structure Ambiguity of HL7 Messages," *Computer Methods and Programs in Biomedicine*, vol. 80 (1), pp. 1 - 15, 2005.

[21] P. M. Valdiguié, E. Rogari, and H. Philippe, "VALAB: Expert System for Validation of Biochemical Data," *Chlinical Chemistry*, vol. 38 (1), pp. 83 - 87, 1992.

[22] W. P. Oosterhuis, H. J. L. M. Ulenkate, and H. M. J. Goldschmidt, "Evaluation of LabRespond, a New Automated Validation System for Clinical Laboratory Test Results," *Clinical Chemistry*, vol. 46 (11), pp. 1811 - 1817, 2000.

[23] G. Boran, P. Given, and R. O'Moore, "Patient Result Validation Services," *Computer Methods and Programs in Biomedicine*, vol. 50 pp. 161 -168, 1996.

[24] J. X. Corberand, "Computer Validation in Hematology," *Immuno-analyse & Biologie spécialisée*, vol. 18 pp. 133 - 137, 2003.

[25] J. C. Libeer, "Validation of Clinical Laboratory Results: Discussion of Essential Validation Elements," *Drug Information Journal*, vol. 31 pp. 243 - 250, 1997.

[26] C. A. Holland and F. L. Kiechle, "Point-of-care molecular diagnostic systems — past, present and future," *Current Opinion in Microbiology*, vol. 8 (5), 2005.

[27] B. M. Goldsmith, Optimizing Point-of-care Testing, *Advance for Administrators of the Laboratory*, http://laboratory-manager.advanceweb.com/common/Editorial/Editorial.aspx?CC=38611, 2005, Accessed September 2005

[28] C. A. Lehmann, "The Future of Home Testing - Implications for Traditional Laboratories," *Clinica Chimica Acta*, vol. 323 pp. 31 - 36, 2002.

[29] S. Skeie, G. Thue, K. Nerhus, and S. Sandberg, "Instruments for Self-Monitoring of Blood Glucose: Comparisons of Testing Quality Achieved by Patients and a Technician," *Clinical Chemistry*, vol. 48 (994 - 1003), 2002.

[30] R. M. Bergenstal and J. R. Gavin III, "The role of self-monitoring of blood glucose in the care of people with diabetes: report of a global consensus conference," *The American Journal of Medicine*, vol. 118 (1), pp. 1 - 6, 2005.

[31] Diabetes, *Lab Tests Online*, http://www.labtestsonline.org/understanding/conditions/diabetes.html, Accessed October 2005

[32] Ascensia (TM), *Bayer Diagnostics*, http://www.ascensia.co.uk/, 2005, Accessed October 2005

[33] Accu-Check, *Roche diagnostics*, http://www.accu-chek.co.uk/, 2005, Accessed October 2005

[34] One Touch Glucose Meters, *LifeScan*, http://www.lifescan.com/, 2005, Accessed October 2005

[35] Warfarin, *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Warfarin, 2005, Accessed October 2005

[36] F. Newall, P. Monagle, and L. Johnston, "Home INR monitoring of oral anticoagulant therapy in children using the CoaguChek S point-of-care monitor and a robust education program," *Thrombosis Research*, Article in press (Available online15 September 2005) 2005.

[37] HemoSense®, maker of the INRatio® Monitor, *MemoSense*, http://www.hemosense.com/, 2005, Accessed October 2005

[38] CoaguCheck System, *Roche diagnostics*, http://www.coaguchek.co.uk/, 2005, Accessed October 2005

[39] Prothrombin Time, *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/International_normalized_ratio, 2005, Accessed October 2005

[40] S. Josifovska, "Where Next for the Handset?," *IEE Review*, vol. 50 (12), pp. 36 – 39., 2004.

[41] ARM Processors, http://www.arm.com/products/CPUs/index.html, Accessed August 2005

[42] D. Dagon, T. Martin, and T. Starner, "Mobile phones as computing devices: the viruses are coming!," *IEEE Pervasive Computing*, vol. 3 (4), pp. 11 - 15, 2004.

[43] History of GSM, *GSM World – the website of the GSM Association*, http://www.gsmworld.com/about/history/index.shtml, Accessed August 2005

[44] G. Heine, *GSM Networks: Protocols, Terminology and Implementation*, First ed: Artech House Publishers, 1999.

[45] What is GPRS?, *GSM World – the website of the GSM Association*, http://www.gsmworld.com/technology/gprs/intro.shtml, Accessed August 2005

[46] I. Poole, "From Analogue to 3G," *IEE Communications Engineer*, vol. 1 (3), pp. 26 – 29, 2003.

[47] 3G and UMTS, a new and impatient reality, *Bankinter Innovation Foundation, Future Trends Forum*, http://www.ftforum.org/doc/3G_UMTS_A_new_and_impatient_reality.pdf, Accessed August 2005

[48] G. Peersman, S. Cvetkovic, P. Griffiths, and H. Spear, "The Global System for Mobile Communications Short Message Service," *IEEE Personal Communications*, vol. 7 (3), pp. 15 – 23, 2000.

[49] What is WAP?, *GSM World – the website of the GSM Association*, http://www.gsmworld.com/technology/wap/intro.shtml, Accessed August 2005

[50] K. Read and F. Maurer, "Developing Mobile Wireless Applications," *IEEE Internet Computing*, vol. 7 (1), pp. 81-86, 2003.

[51] M. Campione and K. Walrath, About the Java Technology, *Sun Microsystems*, http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html, Accessed September 2005

[52] Java Glossary, *Sun Microsystems*, http://java.sun.com/docs/glossary.html, Accessed September 2005

[53] R. Riggs, A. Taivalsaari, and M. VandenBrink, *Programming Wireless Devices with the Java 2 Platform, Micro Edition*, 1st ed: Addison-Wesley, 2001.

[54] Building Compelling Services for the Wireless Market Using Java Technology, *Sun Developer Network*, http://developers.sun.com/techtopics/mobility/getstart/articles/whyjava/, 2001, Accessed June 2005

[55] J2ME Devices, *Sun Microsystems*, http://developers.sun.com/techtopics/mobility/device/device, Accessed September 2005

[56] Connected Limited Device Configuration Specification Version 1.1 for Java 2 Micro Edition (JSR-139), *Sun Microsystems*, http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html, 2003, Accessed June 2005

[57] Y. Feng and J. Zhu, *Wireless Java Programming with J2ME*, 1st ed: Sams, 2001.

[58] E. Ortiz, A Survey of J2ME Today, *Sun Developer Network*, http://developers.sun.com/techtopics/mobility/getstart/articles/survey/, October 2004, Accessed November 2005

[59] H. M. Deitel, P. J. Deitel, and S. Santry, *Advanced Java 2 Platform: How To Program*, 1st ed: Prentice-Hall, 2002.

[60] B. Day, Developing Wireless Applications using the Java 2 Platform, Micro Edition, *Sun Developer Network*, http://developers.sun.com/techtopics/mobility/getstart/articles/wirelessdev/, 2001, Accessed April 2005

[61] Mobile Information Device Profile for Java 2 Micro Edition Version 2 (JSR-118), *Sun Microsystems*, http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html, 2002, Accessed June 2005

[62] CORBA Specification, Version 3.03, *Object Management Group*, http://www.omg.org/technology/documents/formal/corba_iiop.htm, 2003, Accessed May 2005

[63] M. Haahr, R. Cunningham, and V. Cahill, "Supporting CORBA Applications in a Mobile Environment," presented at MobiCom '99: 5th International Conference on Mobile Computing and Networking, Seattle, 1999.

[64] *Java 2 Platform, Micro Edition (J2ME) Web Services*: Sun Microsystems, A Technical White Paper, 2004.

[65] C. E. Ortiz, Introduction to J2ME Web Services, *Sun Developer Network*, http://developers.sun.com/techtopics/mobility/apis/articles/wsa/index.html, 2004, Accessed May 2005

[66] V. Chopra, A. Bakore, J. Eaves, B. Galbraith, S. Li, and W. C., *Professional Apache Tomcat 5 (Programmer to Programmer)*, 1st ed: Wiley Publishing, Inc., 2004.

[67] K. Kline and D. Kline, *SQL in a Nutshell*, 1st ed: O'Reilly and Associates, 2001.

[68] J. Knudsen, Networking, User Experience, and Threads, *Sun Developer Network*, http://developers.sun.com/techtopics/mobility/midp/articles/threading/, January 2002, Accessed April 2005

[69] Q. H. Mahmoud, Secure Java MIDP Programming Using HTTPS with MIDP, *Sun Developer Network*, http://developers.sun.com/techtopics/mobility/midp/articles/https/, 2002, Accessed November 2005

[70] Blood Test Results Nomal Range Reference Chart, *Blood Book*, http://www.bloodbook.com/ranges.html, Accessed November 2005