A home based health monitoring system: An implementation and evaluation

Håkon Unander Haugros
Signe Bø Overå

Master of Science in Computer Science

Submission date:	June 2008
Supervisor:	John Krogstie, IDI
Co-supervisor:	Øyvind Strømme, Accenture

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Recently there has been a change in people's demands and expectations of health care systems. Information is easily available through the internet which leads to increased patient knowledge. Our demands are leading to health care that improve quality of life throughout the continuum of life.

Health care monitoring can now be done ubiquitously and there are several different research projects into wearable health monitoring. However all the projects are at a prototype stage and not in wide scale use.

The LifeShirt is a non-invasive continuous monitoring system that collects data on cardiac, pulmonary and other physiological data. The collected data is stored on a memory card or can be transmitted in real time over a wireless network.

This project will build upon earlier work done by us, where we wanted to research the possibility of using the LifeShirt in an integrated health information system, in order to create a health care program that would be more cost efficient or provide new treatment options.

In our earlier work we designed an architecture for a specific user case. In this project we would like to implement the solution. The users of the system, both the involved family and the relevant medical personnel will be available to provide us with feedback on our solution. We can thus test the solution in a close to real situation.

Finally we will evaluate the solution from different perspectives. We will look into the advantages and disadvantages for the patient, as well as look at the quality of the software solution with regards to usability, functionality and future extensibility.

Assignment given: 15. January 2008
Supervisor: John Krogstie, IDI

ABSTRACT

The advances in technology over recent years have opened up a lot of opportunities in the field of wearable health monitoring. Technology equipment that was once reserved for hospital use may now be used in the home of a consumer; this could ease the life of many long term patients and their next of kin.

We have been in contact with the case of a child, who suffers from a number of rare conditions and complications. She has to be monitored almost all the time in her home by her parents. She has been in and out of the hospital a number of times, without ever figuring out what causes her problems.

Her parents have to use a lot of time and effort to monitor her. Our goal in this thesis was two-fold, automate and ease the monitoring of her as well as logging all the data of vital signs so that it may later be used for diagnosing.

We made a prototype system using the hardware of a wearable monitoring shirt called the LifeShirt. Our main focus was to create a system that would allow for discussion around potential usage areas of the LifeShirt. We did live testing on the patient and evaluated the solution with the family and the patient's physician.

We found that our prototype concept fulfills a need that is currently unmet. Their monitoring can be simplified, and the physician can get more data to use for diagnosing purposes.

Based on our result we see a great potential for using wearable health monitoring technology in the home. We envisage many areas that could benefit from automated monitoring with the LifeShirt, both in home as well as in hospital settings.

PREFACE

This document is the report of our Master's Thesis in Computer Science at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU) in Trondheim. This project was a co-operation between IDI at NTNU and Accenture Innovation Lab Norway.

We would like to thank our supervisors John Krogstie from IDI, NTNU and Øyvind Strømme from Accenture. They have given us valuable feedback and council during the work of this thesis. We would also like to thank the chief physician at the pediatrics apartment at a Norwegian hospital, for interesting viewpoints seen from a medical perspective. Least, but not last, we would like to thank the family who has tested our solution and helped us in the evaluation of this thesis.

Trondheim, June 9. 2008

Håkon Haugros

Signe Overå

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

"The greatest wealth is health."
- *Virgil, classical Roman poet*

Development of computer hardware over the last 50 years has followed an exponential curve; this growth rate is likely to continue for some time. Likewise for digital electronic devices, they are ever smaller with an ever increasing performance. Thus enabling new usages and increasing the usefulness of digital electronic devices. Such devices are a driving force for both technological and social changes of our society. There are many examples of devices today that help people live a normal life, even though they have a health complication, for instance a pace maker or a hearing aid.

A trend in later years has been in the development of *ubiquitous* devices. Devices that are present at all times and perform functions that enable and augment humans.
A use of ubiquitous computing is to relieve humans of doing tedious and repetitive tasks. A machine can be set to monitor a situation and under which scenarios it should alert the human, the human is then free to direct his focus elsewhere knowing the situation is under control.

When you are hospitalized today, you are being monitored by computers. Different vital signs in your body are monitored and if anything is abnormal, medical personnel are automatically called. But what if we were able to move this situation to a patient's home instead of in a hospital? What if you could wear a monitoring device in your daily life and know that as soon as something became abnormal; you, your relatives or maybe medical personnel were alerted instantly?

This report describes the implementation, testing and evaluation of a health monitoring system to use at home. Testing has been performed with a real patient and her surroundings, and the evaluation is based on input from this family and involved medical personnel.

1.1 MOTIVATION AND GOAL

A lot of research has been done in the field of health monitoring at home, but except for simple heart rate monitors they have all stopped at the idea or demonstration stage. The main problem is that although monitoring system sounds useful in theory, taking monitoring technology in use in a new environment requires acceptance from all involved parties. Potential users who need monitoring have to rely on the new technology and be comfortable using it. Medical personnel might need to change their working processes, and also be assured that the new technology is better than the existing one.

1.1.1 MOTIVATIONS

The motivation for this thesis was the possibility to create a monitoring system with functionality that has not been developed before, at least not commercially. A system that could be tested on a specific user case that was eager to try out such a system. The medical personnel involved in the specific user case were also positive to such a solution, which meant that they had faith in that the system could improve people's life. Creating a system for one specific case and perhaps make her life and her surroundings better is not something you can do everyday.

The trends in the Norwegian health sector today can be seen as an approach towards a centralized system offering services to different health instances. Our system could be a step further in this direction and this was a motivation as well.

1.1.2 GOALS

The way from demonstration of a health monitoring system and ideas of how it can be used to an actual system in use by medical personnel and patients is quite long. Our goal was to take a monitoring system beyond the demonstration phase and developing an application that could be tested on a specific user case and her surroundings. This could raise discussion and valuable feedback from experience in a real situation could be reported. Testing and evaluation of the system could lead closer to acceptance of this type of monitoring for the stakeholders.

1.2 SCOPE

- The monitoring system was developed to suit one specific use case. It was made for testing and discussion purposes and the functionality was the most important as opposed to the form factor.

- To see if the system could be developed easily modifiable to suit other users was inside our scope. However, developing the system to suit these users and testing the system on others than our specific case was not manageable in this thesis.

- Making the application easy to install and ready for production was not important due to the fact that it was going to be tested on only one user. Again, it was the functionality that was going to be tested and not the installation process.

- Although important, other aspects like for instance security, maintenance and economy of scale were outside our scope. This is important if the system is taken further but not necessary for our testing.

1.3 REPORT OUTLINE

This thesis consists of the following chapters:

Chapter 2 elaborates the problem and describes shortly health monitoring in general. It also describes the monitoring shirt in detail, together with a description of the specific user case. It ends with a brief discussion of the term *health-related quality of life*.

Chapter 3 describes the research questions and the method used to find the answers to these questions. It also states the contributions of this thesis.

Chapter 4 contains the requirements specification. The requirements are outlined as tables and weighted with importance and degree of difficulty. An overall description, functional requirements and quality requirements are described.

Chapter 5 outlines an architecture for the system. It describes architectural drivers and the quality tactics chosen to achieve these. Then an overview of the architecture is presented and described, and in the end the infrastructure are shown.

Chapter 6 describes the design of the system. It starts with a description of the server part through package and class diagrams. Then the clients are described and illustrated. In the end there is a description of how the quality tactics discussed in chapter 5 are fulfilled.

Chapter 7 briefly discusses the implementation phase. It describes choices that were taken and problems that occurred. In the end the requirements which were not implemented are described.

Chapter 8 describes the testing sessions and the results from these. The results are based on the technology acceptance model which is also described here. The chapter ends with a discussion of the reliability of the test results.

Chapter 9 gives a brief description of our contributions. It then summaries and discusses the contributions and the results from testing.

Chapter 10 concludes the report with this thesis' conclusion and discusses further work and possibilities for a personal monitoring system in the future.

2 PROBLEM ELABORATION

"I am dying with the help of too many physicians."
- *Alexander the Great*

With the pervasiveness of technology in the modern society, people expect innovative and integrated solutions. If I have a mobile phone, some health monitoring equipment with alarm functionality and both may connect to the Internet, why shouldn't I be able to connect them so that the alarm shows up on my mobile phone?

In countries where human resources are expensive, technology becomes a driver for change and cost efficiency. Norway, as many of the other countries of the western world, has a high penetration of technology and a population proficient in using technological aids. An example of this is the family which we've developed for, described later on in this chapter. Upon having a disabled child they saw how their current situation was not optimal, and then sought out to improve the situation by using advanced technology. In this case to further automate the monitoring of their child.

Automated monitoring removes the need of constant attention from humans. As literacy limited the need for rote memorization, automated monitoring let caretakers focus their attention elsewhere, to what one can expect is a better use of their energy and time.

Personal health monitoring systems offer many advantages on different levels. One usage area is the one outlined, in a home where monitoring is done by the parent to a child. Another use of the same technology would be between a hospital and elderly or other patients that have a need for monitoring. Since sensor technology and computing technology is becoming ever smaller and better, possible usage areas are ever increasing.

This chapter describes the background information of the current situation. Both in regards to the patient and the current monitoring solution, as well as the current usage and characteristics of the technology used.

2.1 STATE OF THE ART OF PERSONAL HEALTH MONITORING

In this thesis we used the LifeShirt, which is described in the next section, as the hardware device. However, there are similar shirts with similar functionality. Here we briefly outline some of the research into wearable health monitoring.

Wearable health monitoring usually involves a shirt of some kind. A shirt can be worn underneath other clothes and provides a placeholder for sensors and circuitry. Several different shirts have been developed at research labs around the world, though they each have a different focus.

One focus is to integrate the sensors completely into the garment, so that no electrodes have to be placed on the skin, thus making the garment or shirt more accessible and easier to use. Readings from sensors that are woven into the garment is error prone and difficult, however interesting research called the WEALTHY system (Paradiso, 2003) shows this is possible with current technology.

Another interesting focus is Body Area Networks (BAN). A BAN is a short-range network that connects sensors on the shirt or the person together wirelessly. The user usually wears a PDA or a long-range transmitter that sends the signals to a monitoring station. With BANs the sensor architecture becomes more plug-and-play, sensors may be added and removed depending on the usage area. A working prototype called MobiHealth (Jones, Robert, & Istepanian, 2006) demonstrates this principle.

When monitoring becomes long term even a comfortable shirt may prove a nuisance to the user. An approach to solve this is to integrate sensors into the home of the user; this may be used in conjunction with a shirt or as a stand alone system. A team of Japanese researchers created in 1998 a home with integrated sensors in the bed and bathroom (Tamura, 1998). Body temperature was read from a temperature sensor in the bed, and body weight was measured by the toilet. Similar research has been done by a Finnish team in 2003 (Korhonen, Pärkkä, & Van Gils, 2003).

Sensor technology is becoming smaller and smarter. Increase of transistors and consequently increase in computing power on the sensor can remove the need for a computing PDA, energy scavenging can remove the need for battery (Yeatman, 2006), and wireless transfers can remove the need for sensors to be connected by wires to each other or to a PDA.

2.2 ABOUT THE LIFESHIRT

The LifeShirt is a wearable monitoring system developed by VivoMetrics (VivoMetrics, VivoMetrics - About us, 2007). VivoMetrics was founded in 1999 when they got patents for wearable sensors that monitor respiratory and cardiac function. The VivoMetrics team consists of experts in both the medical and engineering field.

VivoMetrics has received both the American Food and Drug Administration (FDA) clearance and clearance from the European Medicines Agency (EMEA) regulatory agency (CE Mark) for the LifeShirt System.

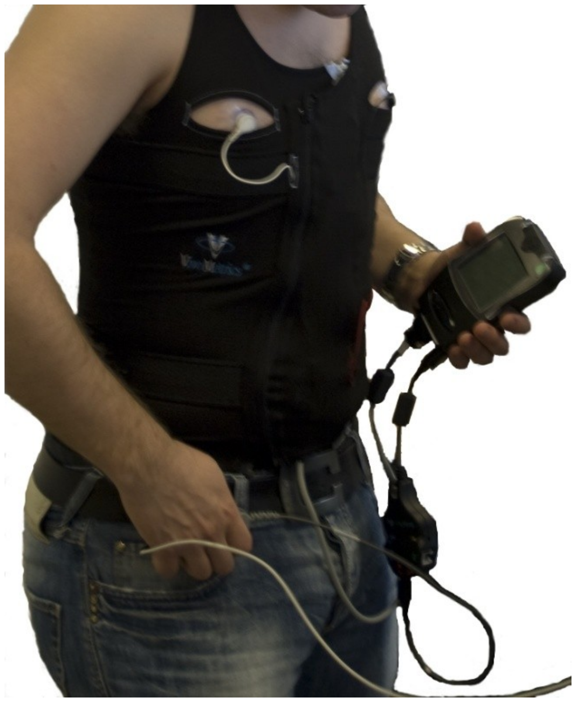


FIGURE 2-1 - THE LIFESHIRT

2.2.1 THE SYSTEM

The LifeShirt is a system that can monitor several different areas of your body. The basic system consists of the wearable shirt itself, a wire with connectors to different sensors, a PDA and a software program. Figure 2-1 shows the LifeShirt with connectors and PDA attached. The wearable shirt comes in all sizes, both for children and adults. This basic system monitors your respiration, heart, motion and position (whether or not you're in an upright position). Respiration is monitored by sensors woven into the shirt around the chest and the abdomen. In addition, it is possible to monitor as much as 30 different values of your body (VivoMetrics, VivoMetrics LifeShirt System Technology, 2004). Among these are blood pressure, blood oxygen saturation and tidal CO_2. You can choose to buy whatever additional equipment you like, which means that each shirt can be specially adjusted for each patient.

The PDA provides input and output. It displays vital signs as well as allows the user to input diary events. A diary event is for instance a registration of when a patient takes some medicine. However, this was not used in our solution.

2.2.2 CURRENT USAGE

VivoMetrics produces two different versions of the LifeShirt, the full LifeShirt and a more lightweight chest strap called the VivoResponder.

The VivoResponder is intended for use in sport and fitness areas or in first responders and biohazard situations. It has embedded sensors for respiration, cardiac function, skin temperature, posture and activity. Along with the software, it enables trainers or support personnel to ensure that monitored subjects are within specified safety parameters.

The full size LifeShirt is used for clinical trials and other research purposes. It may be used on both humans and canines. VivoMetrics' edge is that the LifeShirt provides the same monitoring you may do in a controlled environment even though the study is conducted in an uncontrolled environment, i.e. everyday life.

Both LifeShirt models offer real-time monitoring possibilities as well as post analysis of collected data. The LifeShirt is intended for short term high resolution monitoring, which its software and hardware reflect. For long term monitoring, storage capacity and battery on the LifeShirt are challenges that need to be addressed.

2.2.3 CONNECTING TO THE LIFESHIRT

We have received an API from VivoMetrics which allows us to receive monitoring data from the LifeShirt. Each second we receive a packet with the following data:

- Id of the user
- Timestamp

- Breath rate
- Heart rate
- Motion
- Position
- SpO_2
- Temperature

The API is based on and connects to an older version of the current software. Current software will monitor at a much higher rate, enabling higher resolution graphs. The reason we use an old version is that the API was custom created for another research project, and VivoMetrics does not normally provides API to outsiders.

2.3 OUR SPECIFIC PATIENT CASE

We have explored a specific case about a functionally disabled child. Both her family and her physicians are interested in trying out new solutions that could help her. They both provided valuable input to this project, especially in the requirements for the solution, and in the evaluation of the implemented solution.

2.3.1 PATIENT BACKGROUND

Linda is a 7 year old functionally disabled child. She has respiration problems of unknown origin. These respiration problems occur with varying frequency and in different situations. Since Linda is unable to talk, diagnosing the cause of the respiration problems has proved difficult. In addition it is very difficult to detect any warnings of an apnea or another problem arising.

Linda has been diagnosed with Arnold Chiari syndrome. Arnold Chiari syndrome leads to structural defects in the brain (National Institute of Neurological Disorders and Stroke, 2007). She has gone through several operations, and has currently an intracranial shunt. The shunt is a tube into her brain to extract excess fluid from her brain.

Linda has also been diagnosed with bradycardia; this is when one has a heart beat below 60 beats per minute. This limit is relative, and a heart beat below 60 can be normal for athletes who are in a good shape, however for other people, such a low heart beat signalizes that the heart is unable to pump enough blood to the body. Due to Linda's bradycardia and respiration problems, she has been resuscitated multiple times.

2.3.2 CURRENT SITUATION

Linda is under constant surveillance by her parents. Her mother is a nurse and can therefore provide resuscitation should it be necessary. At night Linda is connected to a SpO_2 monitor that monitors the saturation and pulse. A wireless video camera is installed in Linda's bedroom, it transfers video and audio to a black and white screen so that Linda can be monitored from other places in the home. When the parents are unavailable to take care of Linda, whether it is because of work or as a means of load removal, Linda is at the hospital. There she must be under constant surveillance by a selected group of nurses that are familiar with her case.

2.4 HEALTH-RELATED QUALITY OF LIFE

Within the use of LifeShirt, it makes sense to talk about *health-related quality of life*. When being ill or having a disease; all aspects of life could suddenly become health related (Guyatt, Feeny, & Patrick, 1993). Factors like for instance income, social life and lack of freedom can be affected when you are sick. All these factors affect the patient's quality of life, and thereby the health-related quality of life.

The current solution for Linda demands a lot of attention from her parents. Caring for Linda is time consuming, and they have to be alert and ready to act at a moment's notice. When they do not have the time or energy, Linda has to be admitted to the hospital. This means that she stays in hospital only because she needs to be monitored and looked after. When staying at home, Linda needs to be within sight of at least one of her parents or within sight through the camera.

If an easier solution for home monitoring and caring could be made, Linda would have to be admitted less frequently into the hospital. Thus the family has more time together. Also, wearing the LifeShirt with alarms gives the opportunity for Linda's parents to stay in another location and still get messages as soon as something is abnormal. This gives her parents more freedom. Knowing that anything abnormal triggers an alarm could make the family less concerned. We define this as an *improvement in the health-related quality of life* for her family.

2.5 NOVELTY OF WORK

The LifeShirt has a wide range of sensors and can be used to monitor in several different situations. The LifeShirt is originally intended for usage areas other than what is outlined in this thesis. It was interesting to take the strengths of the LifeShirt and see if it could be used in other areas.

There are several similar solutions to the LifeShirt on the market or as proof of concepts. There are some examples of health monitoring in the home, however the use of technology and the purpose of the monitoring varies. While solutions exist, they are usually just prototypes. In Norway, there has been little activity in the field of monitoring systems for use inside a patient's home.

The novelty in our work was that we built a solution that focused on a specific patient case, and made a system that fills the needs of the patient and her family. With the results from this trial we gained insights which can be used to generalize a larger system for other patients as well.

3 RESEARCH

"Our health
always seems much more valuable
after we lose it."
- *Unknown*

This chapter states the research questions and the research method for this thesis. The chapter ends with a description of this thesis' contributions.

3.1 RESEARCH QUESTIONS

These were the main questions this project was going to focus on and try to answer.

1. Potential LifeShirt users will have individual functional demands for the usage of a LifeShirt solution. Patients with different diagnoses will have different symptoms and requirements. For instance, an alarm trigger for one user is not necessarily the same for another user. The solution has to be customizable.
    a. Will it be possible to extend the LifeShirt system to make it customizable and useful for different users, not implementing a system from scratch to each new user?

2. The system that was implemented was developed for Linda and her family.
    a. With a custom made LifeShirt solution for Linda, what will be the impact on Linda's family?
    b. Will their lives become easier in dealing with their child's handicap?
    c. Will the medical personnel be able to better diagnose and help Linda?
    d. Will there be a distinct improvement over using the standard solution provided with the LifeShirt?

3.2 RESEARCH METHODS

The research method of this thesis was built around the testing and evaluation of a monitoring system. To be able to test and evaluate, the monitoring system needed to be designed and implemented.

The design and implementation were outlined part by part, due to time limitations. That is, the most fundamental parts on the server were implemented first, then the most important views at the clients and in the end refinements were implemented. This assured that the most important functionality was implemented first, and that less important functionality was implemented if there was time for it.

Before the design and implementation could start, an elaboration of the specific user case and the existing monitoring shirt had to be done. A requirement specification was made and rated according to importance based on input from the specific user case. Quality requirements were also made to assure that the right focus was present at design and implementation time. Also, the architecture of the system was outlined based on the requirements and previous work (Haugros & Overå, 2007).

Implementation of the system was performed and made ready to be tested by our specific user case. The test was planned through making test cases that the test persons should try out. After this session was done, a discussion took place together with the test persons and medical personnel. This discussion gave valuable input for evaluation of the system.

The system was then installed in the specific user case's home. The user case tested the system in a real environment and reported back results from this testing. The test results are described in the light of the *technology acceptance model* (TAM) (Bagozzi & Davis, 1992).

Test results gave the fundament for an evaluation and discussion of different aspects of the system.

The result of this thesis is this report describing the work and results, and the system developed in accordance to the requirements. The research method is illustrated in Figure 3-1.

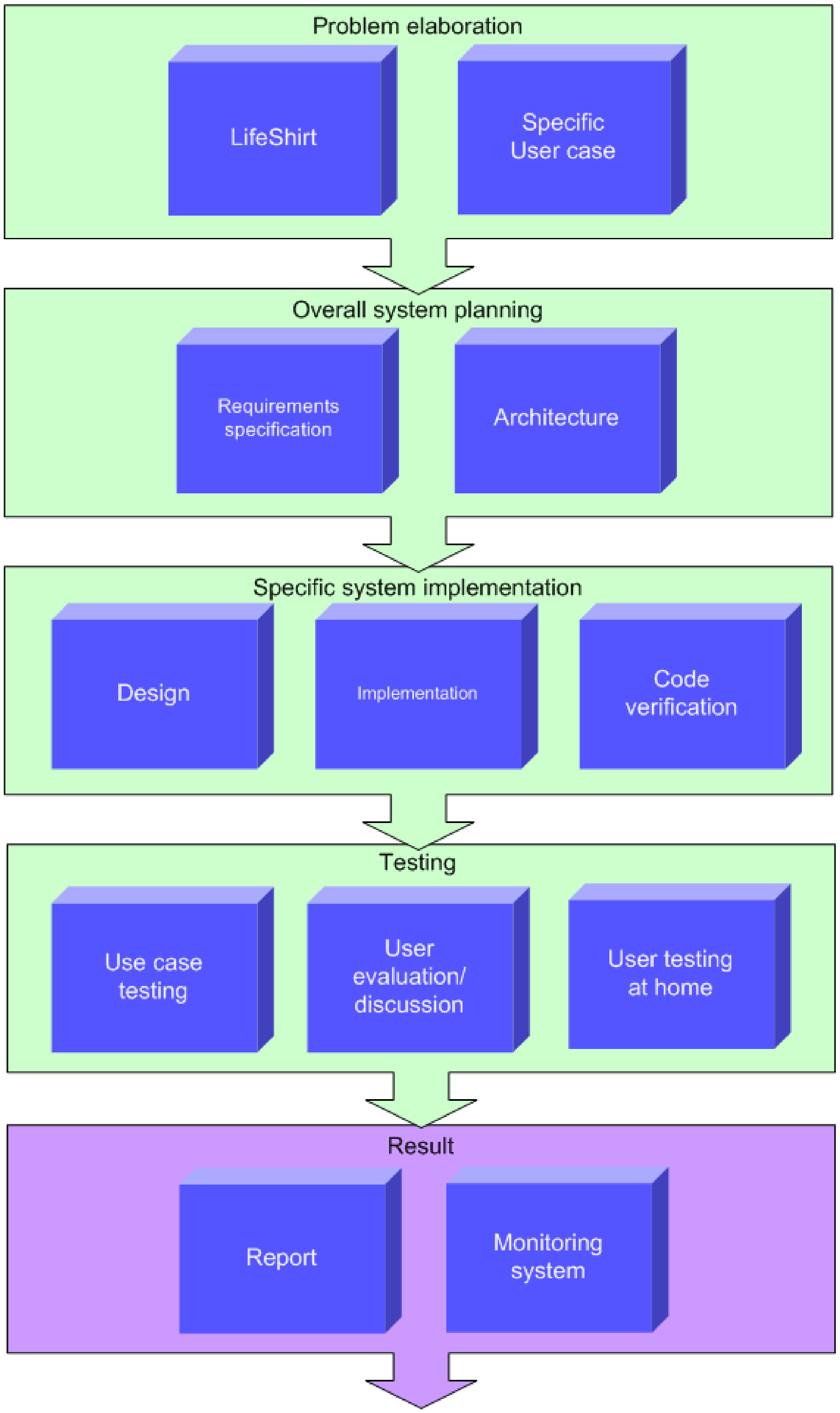


FIGURE 3-1 - RESEARCH METHOD

3.3 CONTRIBUTIONS

The main contribution was the implementation of a system to monitor at home. The server side of the system served as an example of functionality that easily could be extended, and the two clients as examples of how the system could be used by end users. Such a system is not available to buy today.

The thesis also gave an evaluation of a system based on input given from potential users who have tested the system. This discussion could be valuable in future extension of health monitoring system and therefore served as a contribution.

Finally, we suggested future improvements and possible further work.

4 REQUIREMENTS

This chapter presents the requirements for the system. It starts with an overall description of the system and continues with functional and quality requirements.

4.1 OVERALL DESCRIPTION

The system developed is meant to be a monitoring system used together with the LifeShirt. It was developed as three different parts, one server component and two applications; one for mobile clients and one for computer clients.

Monitoring data are transmitted real time through a wireless network card connected to the LifeShirt's PDA. In addition to this, monitoring data are also saved at a memory card in the PDA. Unfortunately data saved in the memory card are not in the same format as the data our application gets real time, and have to be used together with the LifeShirt software VivoLogic.

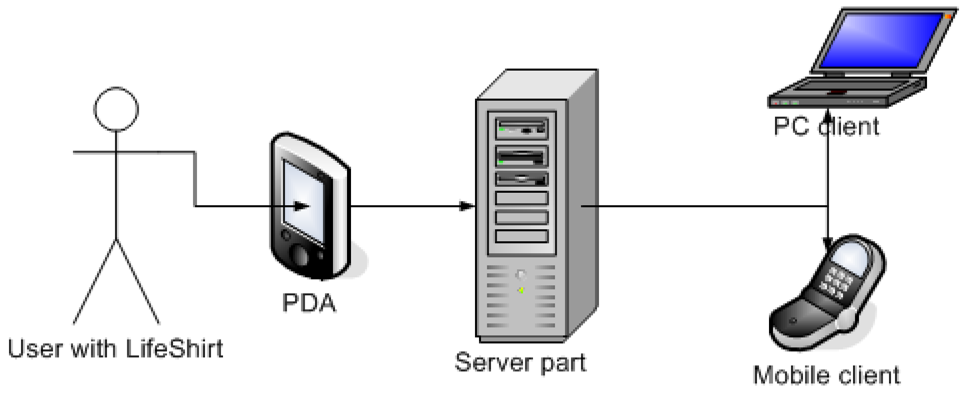


FIGURE 4-1 - DATA FLOW

Figure 4-1 presents the dataflow in the system. Monitoring data are transmitted from the shirt to the PDA attached to the shirt. The PDA sends the data to the server part of the application, which in turn provides services to the clients. A detailed explanation of how the components working and communicate is described in chapter 5 - Architecture.

4.1.1 PRODUCT PERSPECTIVE

The focus was on designing, implementing and testing the system to suit one specific user and her family. This user was described in detail in chapter 2.3. It was also important to try and make the system as modifiable as possible to get a perspective of the possibilities for an extended system.

4.1.2 OPERATING ENVIRONMENT

The monitoring module from LifeShirt was implemented in Windows C++. The server part of the system was implemented in Java 6.0 and the clients in .NET. This required the server as well as the PC clients to run Windows. The mobile application was implemented in .NET Compact Framework 2.0, which requires Windows mobile. Preferably Windows mobile 6, but Windows mobile 5 could also be used.

4.2 FUNCTIONAL REQUIREMENTS

The functional requirements were based on testing of the already existing software, VivoLogic and VivoMonitor, and discussion with Linda's family and doctors. Research done in the fall (Haugros & Overå, 2007) discovered several needs for a monitoring system and these were structured and presented to Linda's family early in this project. Based on a discussion of these requirements a final specification was created.

Having in mind that this was a research project and not a commercial project, the requirements are not a complete list. It contains the most important requirements and the ones that are inside of our scope. They mainly serve as guidelines in design and implementation.

The implementation was executed in several parts, where the most important requirements to make the system work were implemented first. The requirements were therefore weighted according to their importance. These weightings were based on input from Linda's father.

The requirements are presented in a table. Each requirement has a unique id and was weighted by H (high), M (medium) and L (low) importance. They were also weighted with H, M and L according to their degree of difficulty (DOD). This made it easier to decide in which order the requirements should be implemented.

4.2.1 THRESHOLD VALUES

A threshold value means a value that specifies when a vital sign is out of normal range. This is an individual value from user to user. Some might be in critical condition when the heart rate is below 50, but the same might be normal to others.

ID	Description	Importance	DOD
FR 1.1	It should be possible to set a max and a min value for different vital signs, HR, BR and SpO_2	H	L
FR 1.2	Thresholds should be editable on PC client at all time	M	L
FR 1.3	Thresholds should be editable on mobile clients at all times	H	L
FR 1.4	Thresholds should be easily visible in the application	H	L

TABLE 4-1 - THRESHOLD VALUES

4.2.2 MONITORING

The monitoring is the main functionality of the system. Monitoring data is recorded from the LifeShirt and transmitted from the PDA.

ID	Description	Importance	DOD
FR 2.1	Monitoring should send vital signs in real time to PC application	H	M
FR 2.2	Monitoring should send vital signs in real time to mobile application	H	M
FR 2.3	Data should be presented to user as a number, e.g. HR=80	H	L
FR 2.4	Data should be presented on the figure of a person. Vital signs readings should be placed on the anatomically correct part of the figure's body, e.g. the heart rate close to the heart of the figure.	L	M
FR 2.5	The PC client should have a graph view where one can select a time interval for the graphs. The user should be able to specify which vital signs to view and at which interval (e.g. show vital signs from every 5 seconds)	M	H
FR 2.6	Multiple graphs should be possible to view at the same time	L	M
FR 2.7	Monitoring should be possible to perform in different WLANs	M	H
FR 2.8	Vital signs captured should be stored in a central database	L	H
FR 2.9	Vital signs captured should be stored in a database at the PC where server is running	H	L

TABLE 4-2 - MONITORING

4.2.3 ALARM

When something is abnormal in the monitoring, that is if vital signs are out of normal range or there is a sudden loss of connection, an alarm should go off at subscribed listeners. In cases where the user are unable to communicate or watch out for herself, as is the case with the earlier mentioned Linda, alarms automate the monitoring for the patient's caretakers.

ID	Description	Importance	DOD
FR 3.1	When a vital sign breaches a threshold, an alert should be sounded on the PC application	M	M
FR 3.2	When a vital sign breaches a threshold, an alert should be sounded on subscribed mobile clients	H	M
FR 3.3	An alarm should be sounded if connection to the LifeShirt is lost	H	M
FR 3.4	The alarm should be visual	H	L
FR 3.5	The alarm should be audible	H	M
FR 3.6	The alarm should give a visual clue to what triggered the alarm	H	L
FR 3.7	The alarm should give a audible clue to what triggered the alarm	L	H
FR 3.8	Alarms should be possible to manually dismiss	H	M
FR 3.9	Visual alarm should indicate triggered thresholds as long as something is abnormal	H	M
FR 3.10	A sound should be played on mobile clients if connection to server is lost	H	M
FR 3.11	Mobile devices should start vibrating when an alarm occurs	L	M

TABLE 4-3 - ALARM

4.2.4 HELP FUNCTIONALITY

The system could have many potential users with little or no knowledge of medical language or computers. Help functionality was therefore to be added so that users could easily get help where it is needed. It was at this point only planned at the PC application and not at remote clients.

ID	Description	Importance	DOD
FR 4.1	PC application should have in depth explanations of vital signs	L	M
FR 4.2	PC application should have a help function that describes normal interval for vital signs	L	L
FR 4.3	PC application should have step-by-step guides for different program functionality	L	M

TABLE 4-4 - HELP FUNCTIONALITY

4.2.5 REPORTS

The monitoring data could be very interesting to medical personnel involved with the user. Reports generated from monitoring data could for instance be printed and studied as part of the patient’s journal. Reports could help see connections between for instance different events and vital signs.

ID	Description	Importance	DOD
FR 5.1	Data should be searchable based on a start and end date	M	H
FR 5.2	Data should be searchable based on alarms	M	H
FR 5.3	It should be possible to get data from a specified time interval for a trigger sensor around alarm events. It should also be possible to choose other sensors to be presented in the same result	M	H
FR 5.4	Data should be searchable based on vital sign value	M	H
FR 5.5	Stored notes should be displayed as a part of the search result	M	H
FR 5.6	Search results should be presented as graphs	M	M
FR 5.7	Search results should be printable	M	M

TABLE 4-5 - REPORTS

4.2.6 SETTINGS

In different situations it would be useful to have different frequency of monitoring data logging. If the user is in critical condition, higher frequency of the logging would most likely be of interest. On the other hand, there is no use in logging at high frequency when all vital signs are stable and within normal range.

ID	Description	Importance	DOD
FR 6.1	Data should be possible to log at different accuracy. A logging level is defined as a set of parameters that are logged at a given frequency. Different levels may have different parameters	L	H
FR 6.2	It should be possible to have triggers that change the logging level	L	H

TABLE 4-6 - SETTINGS

4.2.7 OTHER REQUIREMENTS

Some additional functionality could be useful to the system, but are not important to make the monitoring work. Creation of notes is introduced here. It could be useful to write notes that could be seen together with monitoring data. This could for instance be what you are doing at the point where something abnormal happens.

ID	Description	Importance	DOD
FR 7.1	In the PC application it should be possible to make notes. These notes are stored together with their time of creation	L	L
FR 7.2	In the mobile client it should be possible to make notes. These notes are stores together with their time of creation	M	L

TABLE 4-7 - OTHER REQUIREMENTS

4.3 QUALITY REQUIREMENTS

This section presents the quality requirements that should be fulfilled by the system. The creation of these was under the assumption that VivoMetrics provides the necessary parts to the system. The requirements are presented as Quality Attribute Scenarios, as suggested by (Bass, Clements, & Kazman, 2006). The scenario tables contain the following parts:

ID: the identification tag of the requirement
Source of stimulus: the entity that generated the stimulus
Stimulus: the condition that needs to be considered when it arrives at the system
Artifact: the pieces of the system that is stimulated
Environment: the condition of the system that the stimulus occurs within
Response: the activity undertaken after the arrival of the stimulus
Response measure: a measure of the response so that the requirement can be tested

The following sections will describe requirements for the quality attributes modifiability, usability, testability and availability. Attributes such as security and performance are left out since they were outside of our scope. However, this does not mean that they are not important in an extended future system.

4.3.1 MODIFIABILITY

The modifiability of the system is about time and cost of changes. Our system was built around one specific family and their requirements. The idea was that the system should be adaptable to other users as well, and this meant that modifiability was very important. It should be possible to change parameters and existing functionality to suit new users, as well as adding new functionality.

Table 4-8 specifies the scenario “A developer wishes to modify the outline of reports”.

Table 4-9 specifies the scenario “A developer wishes to add a new sensor value”.

Table 4-10 specifies the scenario “An end user wishes to modify the threshold values for alarm triggering”

Portion of scenario	Value
Id	QR1
Source	Developer
Stimulus	Wishes to modify the outline of reports
Artifact	Code
Environment	At design time
Response	Modifications are made with no side effects in other parts of the system
Response measure	Modification made within three hours

TABLE 4-8 - QR1

Portion of scenario	Value
Id	QR2
Source	Developer
Stimulus	Wishes to add a new sensor value
Artifact	Code
Environment	At design time
Response	The functionality is added and tested to work with the existing functionality
Response measure	The new build is ready within 5 hours

TABLE 4-9 - QR2

Portion of scenario	Value
Id	QR3
Source	End user
Stimulus	Wishes to modify the threshold values for alarm triggering
Artifact	Functionality
Environment	At runtime
Response	Visual alarm and sound alarm is updated according to the new threshold values
Response measure	The new alarms are active within two seconds

TABLE 4-10 - QR3

4.3.2 USABILITY

Usability is concerned with how easy it is for a user to use the system's functionality and how much support the system provides to the user.

As mentioned earlier the existing software from VivoMetrics is intended for medical personnel, and difficult to take in use for ordinary people. It was therefore important that this system is easy to use and understand.

Table 4-11 specifies the scenario "An end user doesn't understand what the sensors mean".

Table 4-12 specifies the scenario "An end user wants to minimize impact of errors"

Portion of scenario	Value
Id	QR4
Source	End user
Stimulus	Doesn't understand what the sensors mean
Artifact	System
Environment	At runtime
Response	Looks up in help function to read about the sensors
Response measure	Knowledge about sensors are found within five seconds

TABLE 4-11 - QR4

Portion of scenario	Value
Id	QR5
Source	End user
Stimulus	Minimize impact of errors
Artifact	System
Environment	At runtime
Response	Wishes to cancel current operation
Response measure	Cancellation takes less than one second

TABLE 4-12 - QR6

4.3.3 TESTABILITY

The testability of software says something about how easy it is to discover faults through testing. Making a system testable means creating a system where it is possible to control internal state and input, and observe output to see if it is as expected.

The system should be used in situations where potential life danger could appear. A minor error could be fatal. The system was built in a way that makes it easy to discover faults early, especially functionality related to monitoring.

The system was implemented in parts, and each part was properly tested before the next one started.

Table 4-13 specifies the scenario “The developer has completed one iteration and needs to test it”.

Portion of scenario	Value
Id	QR6
Source	The developer
Stimulus	One iteration complete
Artifact	Build
Environment	At development time
Response	The software gives computed values or informative error messages
Response measure	All faults should be masked and no failures occur

TABLE 4-13 - QR7

4.3.4 AVAILABILITY

Availability is concerned with preventing system failure. A failure is observable by the user and means that the system no longer delivers the requested services. A fault that is not corrected or masked often leads to a failure.

The system should discover life threatening conditions and if it fails frequently there is no meaning in using the system. The users need to be able to trust the system.

Table 4-14 specifies the scenario “An unexpected message arrives from LifeShirt”.

Portion of scenario	Value
Id	QR7
Source	External
Stimulus	Unexpected message arrives from LifeShirt
Artifact	Communication channel
Environment	Normal operation
Response	Log message and continue to operate
Response measure	No downtime

TABLE 4-14 - QR8

5 ARCHITECTURE

This chapter describes the architecture of the system and the rationale for it. The system should be used together with the LifeShirt and the architecture is based on requirements from chapter 4.

The architectural drivers will be described in the beginning. These are the most important quality requirements to consider in creating the architecture. Thereafter the quality tactics to achieve the quality requirements are presented. These are tactics that applies both in the architectural description and in the design and implementation phase.

The architecture is then presented through two different views to increase the understanding.

5.1 ARCHITECTURAL DRIVERS

The requirements in chapter 4 contain the functional and quality requirements that should have been fulfilled in the implementation. The ones that were considered most important are called the architectural drivers. There were several qualities that were relevant for the system, and the most important were *modifiability* and *availability*. Although these were in focus it was important to assure that the other qualities mentioned earlier were not forgotten. The modifiability should for instance not be in the way of achieving high usability.

Modifiability was of high importance since every potential user of the system has different symptoms and diseases that they want to monitor. Therefore the system should be modifiable in a way so that the system can be used by several different users and user groups. Availability was of high importance because life threatening situations can occur to the user, and if the system is down it can not notify users of the event. The system's meaning would then be gone.

5.2 QUALITY TACTICS

A quality tactic can be seen as an architectural guidance for achieving the architectural drivers. The tactics are used by the architect to create a design. How these tactics were implemented is described in section 6.4 - Tactics consistence.

5.2.1 MODIFIABILITY TACTICS

Modifiability is about the time and cost to implement, test and deploy changes. Modifiability tactics helps making changes as time and cost efficient as possible.

Tactics for improvement of modifiability can be divided into three classes:

- Localize modifications
- Prevent ripple effects
- Defer binding time

5.2.1.1 LOCALIZE MODIFICATIONS

In general, having a change affect as few modules as possible will reduce the cost and time needed to do the change. The tactic chosen to achieve this in the system was *maintaining semantic coherence*. The goal is that each module is as self contained as possible. Achieving this could be done by defining responsibilities for each module so that they are semantically coherent.

The system's design tried to abstract common services with for instance the use of middleware. This leads to different modules not needing services directly from each other. This tactic also prevents ripple effects.

5.2.1.2 PREVENT RIPPLE EFFECTS

Ripple effects occur when a modification to one module requires modification to other modules as well, only because they depend on the original modification module. A module can depend on another in several ways.

It was used two tactics for achieving this in the system, *hide information* and *maintain existing interfaces*. Hiding information means making as much data as possible private to each module, so that changes in these data don't have an effect on other modules. Maintaining existing interfaces is a tactic where the focus is on letting interfaces be unchanged and only change the classes that is build on the interface. This means that modules using the interface will not take notice of the change. This is not possible in cases where new services are added, but in such cases you could for instance create an additional interface.

5.2.1.3 DEFER BINDING TIME

These types of tactics are related to late binding of parameters. In the system most parameters are possible to set through the user interface, and the binding will happen when the user gives input.

5.2.2 AVAILABILITY TACTICS

Availability tactics helps preventing faults to become failures and make repair of faults possible. A failure occurs when the system no longer delivers the requested services and is visible to the user. A fault could lead to a potential failure.

Tactics for improvements of availability can be divided into three classes:

- Fault detection

- Fault recovery
- Fault prevention

5.2.2.1 FAULT DETECTION

Ping/echo was used to detect connection faults from the mobile clients to the server. This was done to assure that the mobile client has connection with the server and that the server is up and running. If the server connection is lost, the user on a mobile client is notified quickly.

To detect other faults on both the server side and client side, exceptions were used. Exception is raised whenever a fault occurs, and all exception are handled so that a failure does not occur.

5.2.2.2 FAULT RECOVERY

A critical failure occurs when the alarms for some reason doesn't work. There was no need to restore the system or use rollback, and no monitoring data is lost due to the fact that they are stored in the memory card attached to the PDA as well. It could have been of interest to use some kind of redundancy to assure that the services continues to operate even if one process creates a failure. However, this was too complex to set up at this point, especially since the system was made for testing at one particular user, though it should be considered in a future system.

5.2.2.3 FAULT PREVENTION

To prevent faults, transactions were used. This means that all parts of a transaction either are performed, or not. If a transaction fails before finished, earlier steps are undone. This ensured that the system is in a stable mode at all time.

5.3 ARCHITECTURAL VIEWS

This section presents an overview of the architecture through two different views, logical view and infrastructure view. Each of the main components in the two views is described.

5.3.1 LOGICAL VIEW

The solution was divided into four separate areas; Clients, Server, Hardware and Network. There are two different clients that connect to the server, the PC application and the more lightweight mobile client. The server layer receives data from hardware components. These data are processed and stored. This layer exposes web services to clients. Figure 5-1 shows the high level architecture.

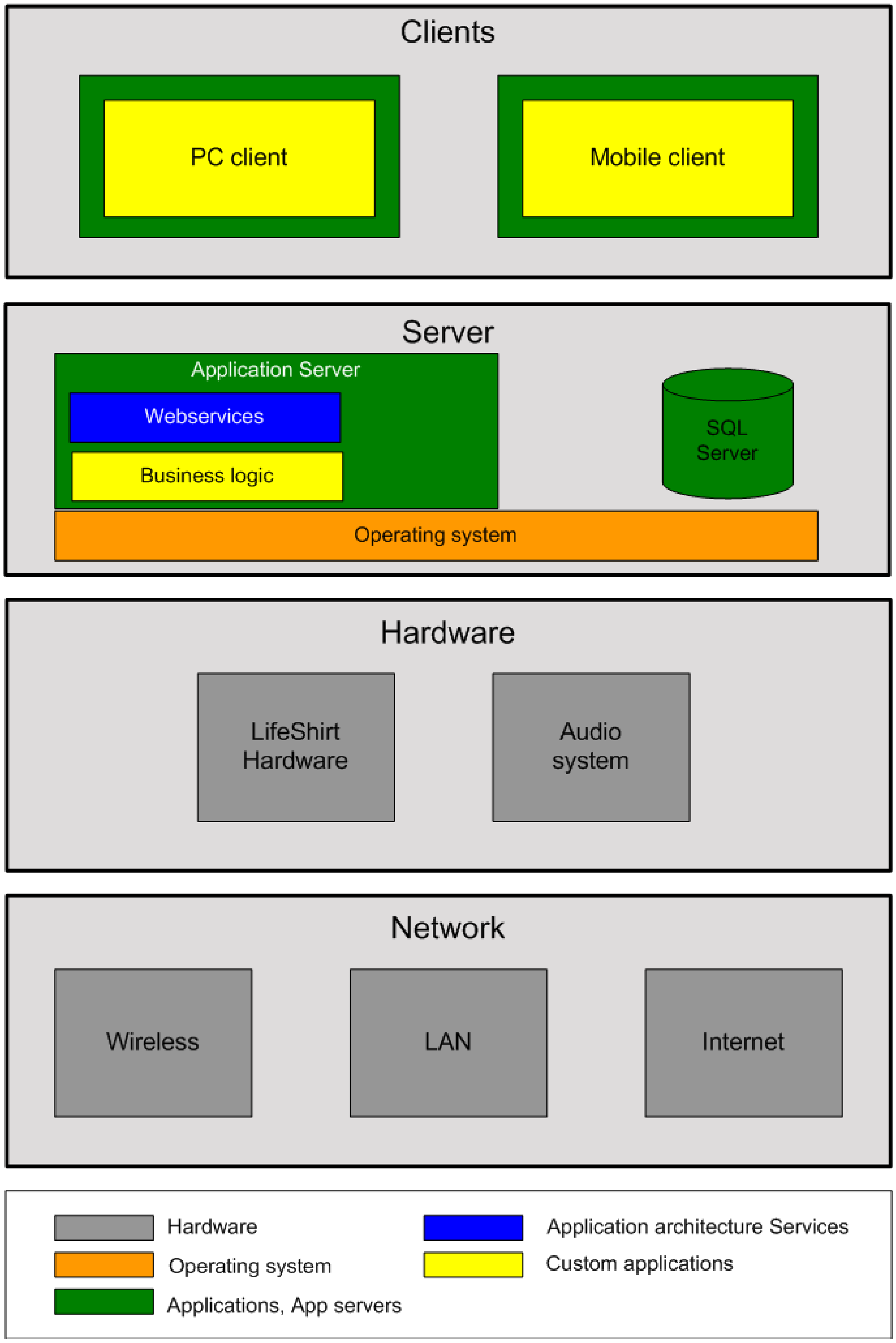


FIGURE 5-1 - LOGICAL OVERVIEW

5.3.1.1 CLIENTS

There are two clients, the PC client and the mobile client. The PC client functions as a view for the monitoring and an alarm central. The PC client runs on the same computer as the server part. It, unlike the mobile client, provides graph capabilities.

The mobile client is a lightweight version of the PC client made for mobile devices. It provides an easy and more portable access to the monitoring. It has an alarm system similar to the PC client. The mobile client can access the server from a remote location via the Internet.

5.3.1.2 SERVER

The server receives monitoring data from the LifeShirt. It stores the monitoring data in a SQL database. The server has multiple web services that are used to interface with clients. Subscribed clients receive data that the server publishes, such as monitoring data to display on the clients.

The server has logic that enables it to analyze the data and trigger an alarm based on current settings. Settings are sent to the server via the PC client. The server also has logic to fetch a certain amount of monitoring data to generate graphs.

5.3.1.3 HARDWARE

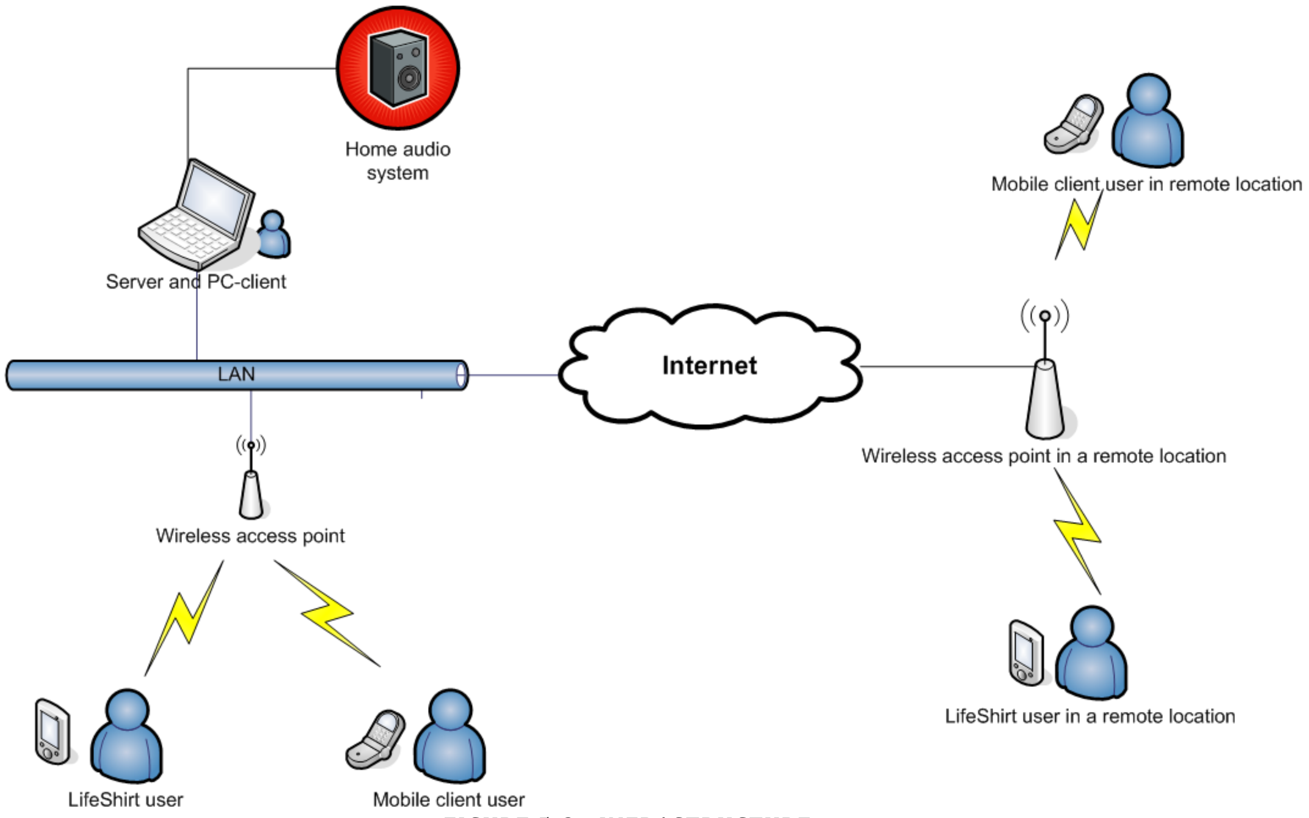
The server must have adequate memory and processing power. A standard laptop with 1GB RAM will be sufficient. The system should be connected to an audio system to provide audible alarms. It must also be connected to the LifeShirt through a network.

5.3.1.4 NETWORK

The server should be connected to a local area network (LAN). This LAN should have a wireless access point that the LifeShirt connects to. This LAN should also be connected to the Internet.

5.3.2 INFRASTRUCTURE VIEW

Figure 5-2 presents the infrastructure for the system.


Home audio system
Server and PC-client
Mobile client user in remote location
LAN
Internet
Wireless access point
Wireless access point in a remote location
LifeShirt user
Mobile client user
LifeShirt user in a remote location
FIGURE 5-2 - INFRASTRUCTURE

5.3.2.1 NETWORK

An Ethernet LAN should be connected to a wireless access point, providing both wired and wireless transfer. If one wireless access point does not provide sufficient coverage, multiple wireless access points could be installed. The LAN could be connected to the Internet, providing remote access.

5.3.2.2 SERVER AND PC-CLIENT

The server and PC-client should run on the same computer. It receives the information from the LifeShirt and is the hub that mobile clients connect to. It is connected to an audio system in order to provide audible alarms in the home location.

5.3.2.3 LIFESHIRT USER

At installation the LifeShirt is configured with the server's IP address. Afterwards every time the LifeShirt is started it will automatically send its monitoring data to the server.

The LifeShirt has to use a different configuration at the remote location, in order to connect to the right wireless access point. In order to maintain two different configurations, the LifeShirt could use two different memory cards, one for each location. Thus making it easy to use the LifeShirt in different locations.

5.3.2.4 MOBILE CLIENT

The mobile client must have wireless LAN (WLAN) capabilities, and should connect to the server through this WLAN.

The mobile client could be in a remote location, normally the same location as the LifeShirt user is in, however all data are first sent to the server for processing and then back to the mobile client.

6 DESIGN

This chapter contains the design of the applications in the system. The first part covers the design of the server side of the application, followed by the front end PC application and the mobile application. In the end there is a description of how the quality tactics discussed in chapter 5 were fulfilled.

The design phase was performed in parts together with the implementation due to time restrictions. The most important functionalities according to the requirement specification in chapter 4 were prioritized. The design was built in a way that makes it easy to add functionality in the future.

6.1 SERVER SIDE

The server side consists of a data access layer and a business logic layer. Figure 6-1 presents the server side modules.

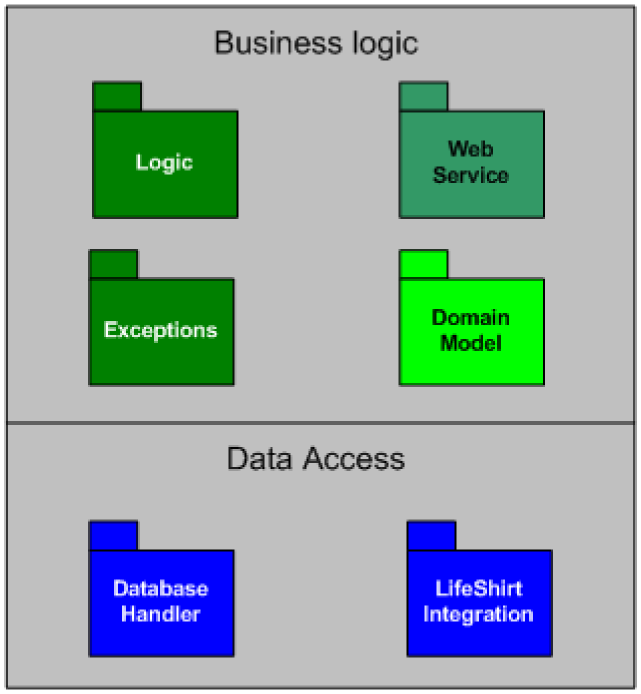


FIGURE 6-1 - SERVER PACKAGES

6.1.1 DATA ACCESS

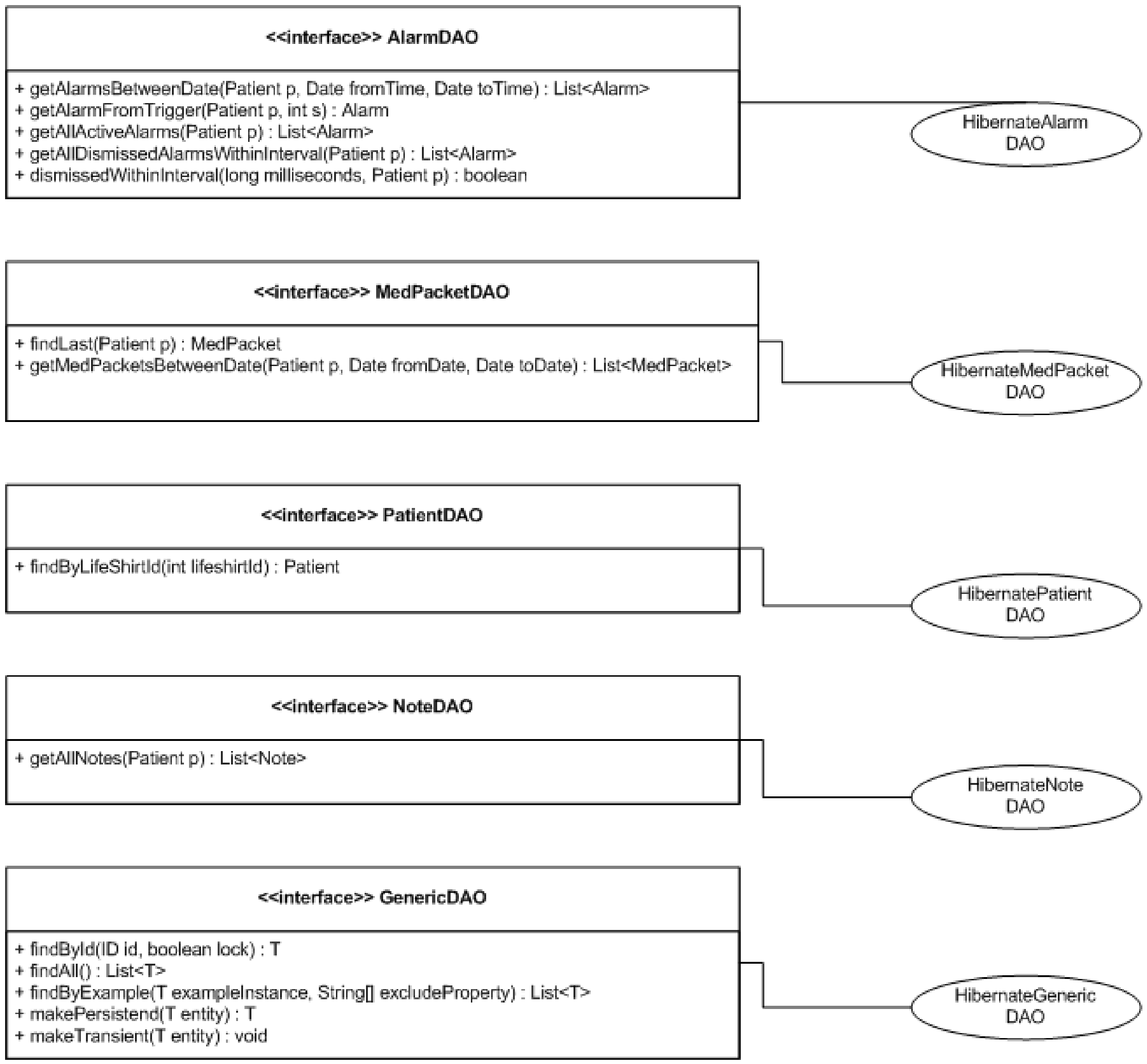
The data access layer handles storage and fetching of data. Data is stored in a database and for a detailed description of this database schema, see appendix A.

6.1.1.1 LIFESHIRT INTEGRATION

Packets with the sensor values are received from the LifeShirt through the software provided by VivoMetrics. These packets are processed from text to objects and stored in the database. The PDA connected to the LifeShirt requires the user to insert a code before the monitoring starts. This code has to be the same as a code inserted in the user interface on the PC client. The code connects the medical packets to the user.

6.1.1.2 DATABASE HANDLER

The database access objects (DAO) are shown in Figure 6-2. Hibernate is used for database access. The use of Hibernate assures that for instance the domain model and the database schema always are consistent. The DAOs have interfaces where new methods are added, and also one interface in common where general methods are placed.


<<interface>> AlarmDAO
+ getAlarmsBetweenDate(Patient p, Date fromTime, Date toTime) : List<Alarm>
+ getAlarmFromTrigger(Patient p, int s) : Alarm
+ getAllActiveAlarms(Patient p) : List<Alarm>
+ getAllDismissedAlarmsWithinInterval(Patient p) : List<Alarm>
+ dismissedWithinInterval(long milliseconds, Patient p) : boolean
HibernateAlarm DAO
<<interface>> MedPacketDAO
+ findLast(Patient p) : MedPacket
+ getMedPacketsBetweenDate(Patient p, Date fromDate, Date toDate) : List<MedPacket>
HibernateMedPacket DAO
<<interface>> PatientDAO
+ findByLifeShirtId(int lifeshirtId) : Patient
HibernatePatient DAO
<<interface>> NoteDAO
+ getAllNotes(Patient p) : List<Note>
HibernateNote DAO
<<interface>> GenericDAO
+ findById(ID id, boolean lock) : T
+ findAll() : List<T>
+ findByExample(T exampleInstance, String[] excludeProperty) : List<T>
+ makePersistend(T entity) : T
+ makeTransient(T entity) : void
HibernateGeneric DAO
FIGURE 6-2 - DATABASE ACCESS OBJECTS

6.1.2 BUSINESS LOGIC

Figure 6-3 shows the business logic for the server application.

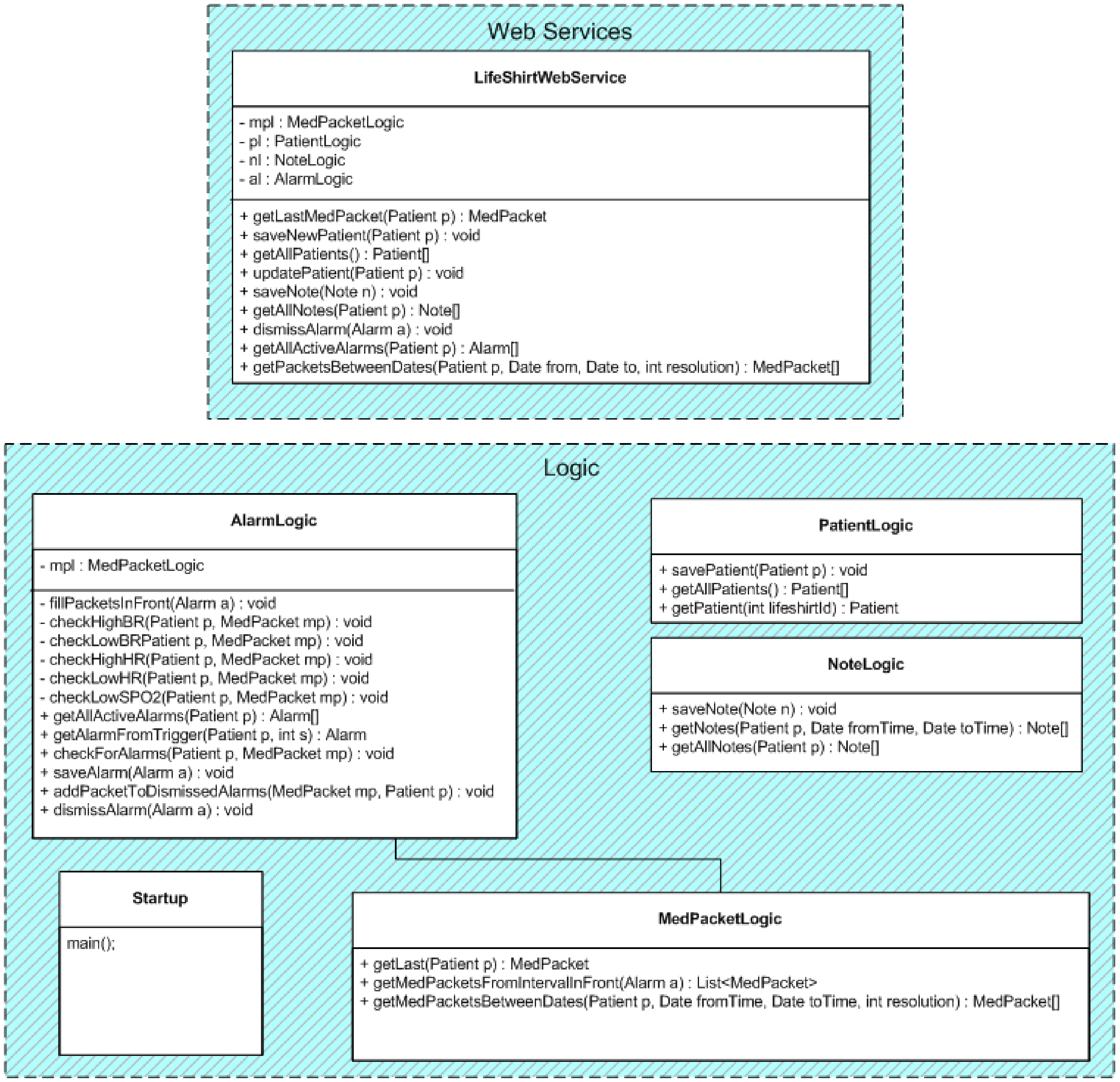

Web Services
LifeShirtWebService
- mpl : MedPacketLogic
- pl : PatientLogic
- nl : NoteLogic
- al : AlarmLogic
+ getLastMedPacket(Patient p) : MedPacket
+ saveNewPatient(Patient p) : void
+ getAllPatients() : Patient[]
+ updatePatient(Patient p) : void
+ saveNote(Note n) : void
+ getAllNotes(Patient p) : Note[]
+ dismissAlarm(Alarm a) : void
+ getAllActiveAlarms(Patient p) : Alarm[]
+ getPacketsBetweenDates(Patient p, Date from, Date to, int resolution) : MedPacket[]
Logic
AlarmLogic
- mpl : MedPacketLogic
- fillPacketsInFront(Alarm a) : void
- checkHighBR(Patient p, MedPacket mp) : void
- checkLowBRPatient p, MedPacket mp) : void
- checkHighHR(Patient p, MedPacket mp) : void
- checkLowHR(Patient p, MedPacket mp) : void
- checkLowSPO2(Patient p, MedPacket mp) : void
+ getAllActiveAlarms(Patient p) : Alarm[]
+ getAlarmFromTrigger(Patient p, int s) : Alarm
+ checkForAlarms(Patient p, MedPacket mp) : void
+ saveAlarm(Alarm a) : void
+ addPacketToDismissedAlarms(MedPacket mp, Patient p) : void
+ dismissAlarm(Alarm a) : void
PatientLogic
+ savePatient(Patient p) : void
+ getAllPatients() : Patient[]
+ getPatient(int lifeshirtId) : Patient
NoteLogic
+ saveNote(Note n) : void
+ getNotes(Patient p, Date fromTime, Date toTime) : Note[]
+ getAllNotes(Patient p) : Note[]
Startup
main();
MedPacketLogic
+ getLast(Patient p) : MedPacket
+ getMedPacketsFromIntervalInFront(Alarm a) : List<MedPacket>
+ getMedPacketsBetweenDates(Patient p, Date fromTime, Date toTime, int resolution) : MedPacket[]

FIGURE 6-3 - BUSINESS LOGIC

6.1.2.1 WEB SERVICES

The server side provides an interface for clients to connect to via a web service. The web service methods cover the necessary methods that clients need. Although the web service was made in Java, you can in theory connect to it with any language you want. This makes it possible to create different clients without having to change the server side of the application. Figure 6-3 shows the web service methods.

6.1.2.2 LOGIC

The logic consists of one class for each class in the domain model. This means that for instance logic for alarm handling is found in the AlarmLogic class. Splitting the logic in these classes makes it easier to locate errors and also to add new functionality at a later point. Figure 6-3 presents the methods provided by each logic class.

6.1.2.3 STARTUP

The startup class presented in Figure 6-3 contains a main method which starts the necessary software from VivoMetrics to start monitoring.

6.1.2.4 DOMAIN MODEL

The domain model for the server side of the application is of a set of entity classes that encapsulate the required business concepts. The domain model is presented in Figure 6-4. They have no methods other than getters and setters. The idea was to have the classes in the business logic operate on the classes in the domain model. This way, the classes' domain model could be easily sent between the business logic and the data access layer.

The domain model is also coded inside the wsdl file in the web service which means that the clients use the same objects.

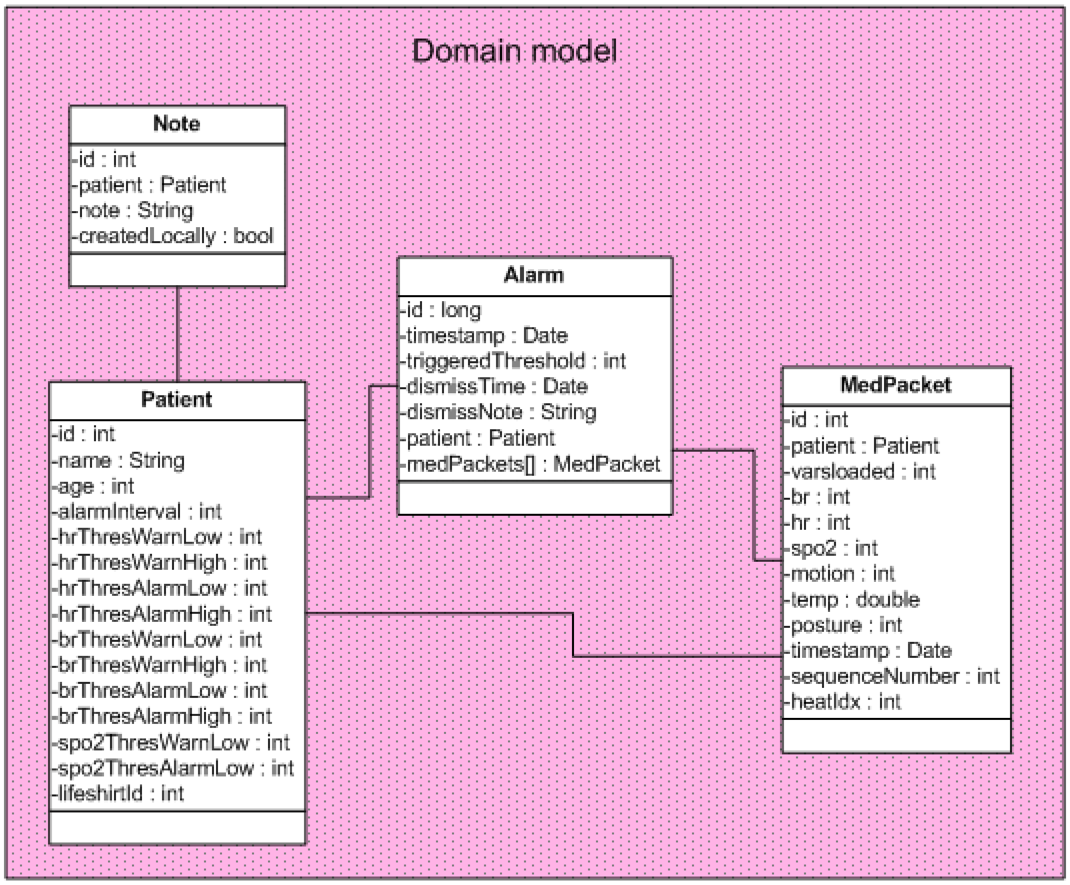


FIGURE 6-4 - DOMAIN MODEL

6.1.2.5 EXCEPTIONS

Four different types of exceptions have been defined; DatabaseException, LifeShirtException, GeneralException and IllegalInputException. DatabaseException is thrown when something goes wrong in communicating with the database. This means whenever Hibernate is unable to store or fetch requested data. LifeShirtException is thrown when the processing of medpackets from LifeShirt fails, for instance if the packet contains unknown text. GeneralException is thrown when something unexpected goes wrong. This covers all other exceptions and assures that the server side won't crash if something unexpected occurs. Then there is the IllegalInputException, which is thrown if input to the web service methods is invalid. This happens when you for instance try to save a patient, and the patient has value *null*.

6.2 PC APPLICATION

The PC client connects to the server at startup via web services. If no connection is achieved it displays the connection error. Upon having a connection the PC client will download the list of patients stored at the server, the user may then select a patient and whether to monitor this patient actively.

The PC client contains functionality for creating and editing patients on the server. When storing a new or edited patient the client will store the information to the server, if there are connection problems the user is alerted to these and the change is not completed. Thus the state between the client and server remains consistent.

When the PC client is set to monitoring, it will actively poll the server for a new medpacket each second. The medpacket is then displayed in the main window. When querying for a medpacket the client will also ask for a list of active alarms from the server, these alarms are then processed in the form of an alert to the user. The alarm is full screen with an active animation and an audible part to alert the patient even if he's not paying attention to the computer. The application will generate an alarm if it doesn't receive any medpackets for 30 seconds; after this alarm is dismissed the PC client will be in an inactive state.

There are two different forms of alarms, one is the *warning* and the other is the *alarm* level. These two levels have different full screen animations as well as sounds. Should an *alarm* occur when a *warning* is active; the alarm window will update itself to display the current situation.

The polling and processing of medpackets and alarms is done in a background worker thread, the main window listens to the worker thread and is alerted when something happens. Thus the GUI may change depending on events from the server. These events are mainly updating the alarms situation, for instance promoting from warning to alarm or when alarms are dismissed from another client.

The client has functionality for creating notes on a patient; this is a simple textbox where a user may input something with a timestamp. The information is then stored on the server.

There is a graph view where a user may inspect the change of vital signs over time. The user inputs which vital signs he would like to inspect and the time interval he is interested in. The user can select which resolution he is interested in. The default value is 1Hz, each second, but the user may opt for each 5 minutes. This is useful for faster generating of graphs when the time interval increases.

Figure 6-5 displays the sequence of steps performed when querying the server. Upon starting monitoring the main form will launch a background worker thread that continuously queries the server. The call is passed through the WsLogic class, which encapsulates the web services methods. The MainForm and the Worker thread works in parallel. In Figure 6-5 the worker thread receives first an alarm list that contains a *warning*. After some time (the dotted line), the worker receives a new alarm list with an *alarm*. The GUI is alerted and updates itself.

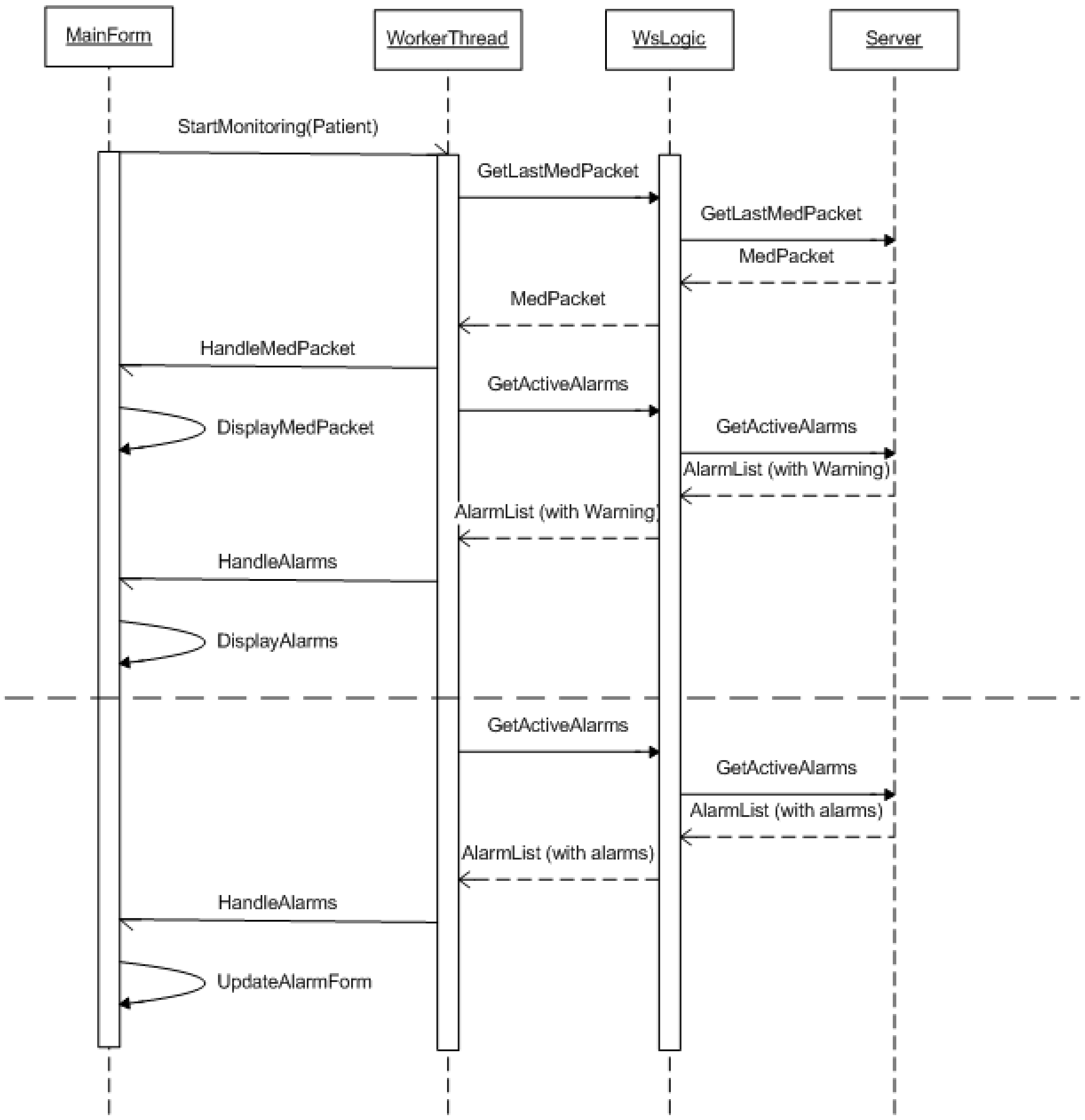


FIGURE 6-5 - SEQUENCE DIAGRAM FOR PC CLIENT

6.3 MOBILE APPLICATION

The mobile application contains parts from the PC application which is useful and possible to have on a mobile device. The mobile application is divided into different forms as shown in Figure 6-6. In addition to the forms there is a logic class which has methods to call each of the web service methods that the server provides. This class is used by the main form.

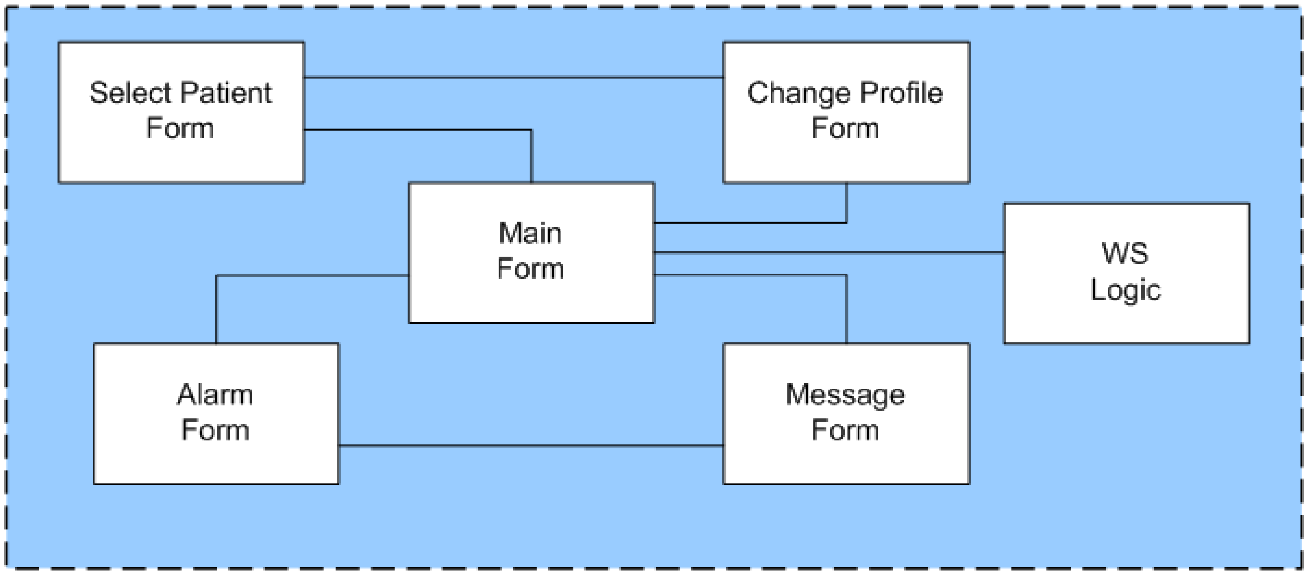


FIGURE 6-6 - MOBILE CLIENT FORMS

The mobile application is designed as clean and simple as possible to make it work on several different phones. The application contains nearly no logic and mainly its job is to present data to the user. Some input are taken from the user, and this is stored at the server side.

6.3.1 MAIN FORM

The whole application is controlled from the main form. It contains a timer which asks the server for medpackets and controls possible alarms every fifth second. The monitoring data are presented in this form. It also contains the main menu where the user's choices are present. When something happens in the timer method or when a user clicks on a choice in the menu, this form controls what other form should be shown. Whenever another form is finished doing the requested work, the focus goes back to this form.

6.3.2 SELECT PATIENT FORM

This form presents all the patients saved in the system and gives the user an opportunity to choose one of them. The user chosen will be the user monitored.

6.3.3 CHANGE PROFILE FORM

This form presents the profile and gives the user opportunity to change the threshold values saved in the user information, or other information about the user. Whenever something is changed this is stored at the server side.

6.3.4 MESSAGE FORM

This form displays a message to the user whenever something goes wrong in the application. The exception handling happens in the main form and the main form uses this message form to inform the user about it.

6.3.5 ALARM FORM

The alarm form displays an alarm. This is done in a manner that makes it easy to discover by the user. The form describes why an alarm went off, and it is possible for the user to dismiss it.

6.4 TACTICS CONSISTENCE

This section describes how the tactics discussed in chapter 5 were fulfilled.

6.4.1 MODIFIABILITY

As shown in Figure 6-3, a lot of methods in the logic classes were made private. This *hides information* from other modules and a change in these would not affect other classes directly. The web service methods were also created with no logic, as a type of interface. A change in the logic will therefore not affect clients using the web service and in this way *the interfaces are maintained.* Also, all the data access objects contain interfaces to hide changes in the actual methods.

6.4.2 AVAILABILITY

Clients are polling the server side for information each second. This is used as the *ping/echo*. If no answer is retrieved within a certain time, the client assumes that the connection to the server is lost. *Exceptions* are used and were described in section 6.1.2.5. In addition to exceptions, unusual behavior and information are *logged*.

Use of Hibernate automatically introduces the use of *transactions*. Whenever something is going to be stored or fetched from the database a transaction is set up and if something crashes in the middle of the transaction, Hibernate makes sure that the database still is in a persistent state. This *prevents faults*.

7 IMPLEMENTATION

As mentioned earlier, development has been done in parts. Whenever new functionality was added, this was done to both the design and the implementation. Verification testing was performed before adding new functionality. This chapter describes what was implemented and the order of it, and how this affected the system.

For an installation guide, see the appendix B. Javadoc has been made for the server component and can be found here[1]. The two clients developed are just examples of clients that can be made and are not documented in the same way as the server part.

7.1 CONNECTING TO THE LIFESHIRT

First the focus was on connecting with our own software to the LifeShirt, using the API provided by VivoMetrics. The API provided had a server component that our software connects to. Data flows at 1HZ into our Java program.

There were several problems integrating the API. The API was written in Windows MFC C++, and is slightly outdated. The API uses native methods of passing messages between programs. We were unable to create our own program that intercepts these messages. We tried loading the library into a managed .Net environment, and creating our own MFC C++ client. However all attempts lead to corrupt memory in libraries out of our control. We therefore decided to modify and use the sample program from VivoMetrics called MsgClient. This program outputs changes to a file, and our program listens for changes to the given file. The current form of integrating with the LifeShirt limits the solution to one concurrent LifeShirt user.

This is a less than optimal solution; in a future version the server part of our application should receive the messages directly, not through an extra program layer.

7.2 SERVER

We proceeded to make a server that receives and processes the medpackets. Medpackets are persisted and linked to a patient. The server also has web services so that clients may connect to it. We made a simple client that connects to the server and polls for medpackets.

Since our server layer was developed in Java, and the clients in .Net we had to use a custom web service configuration to ensure that everything works. The web service generation engine we used is not yet compatible with new features in Java 1.5 and 6, and this requires use of for instance arrays instead of lists in the web service methods and the domain model. This might look a bit outdated, but the functionality remains the same.

[1] http://org.ntnu.no/lifeshirt/

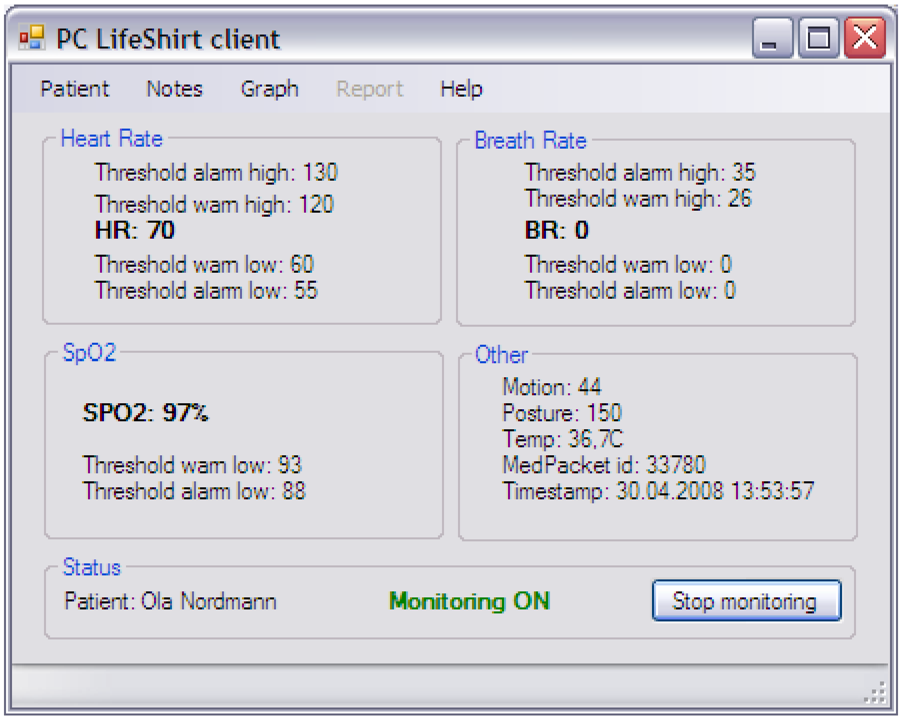
The style of the xml that's transmitted is *RPC/encoded* which was the only style we were able to get to work.

The necessary logic for administrating medpackets, alarms and notes in the server was implemented without any issues. Exception handling was important for the robustness of the system. Exceptions were added and set up to be thrown to clients as soon as something abnormal happens.

7.3 PC CLIENT

The simple PC client created earlier was extended with logic to allow monitoring with alarms. Alarms are generated server side so that medpackets in a given interval around the alarm timestamp may be stored together with the alarm. The client actively polls for new alarms each time it polls for medpackets.

The PC client displays the medpacket information it receives in its main window, see Figure 7-1. When an alarm is received the PC client takes the focus from windows and displays itself in full screen mode, see Figure 7-2. A sound is played in loop to attract user attention.



Threshold alarm high: 130
Threshold warn high: 120
HR: 70
Threshold warn low: 60
Threshold alarm low: 55

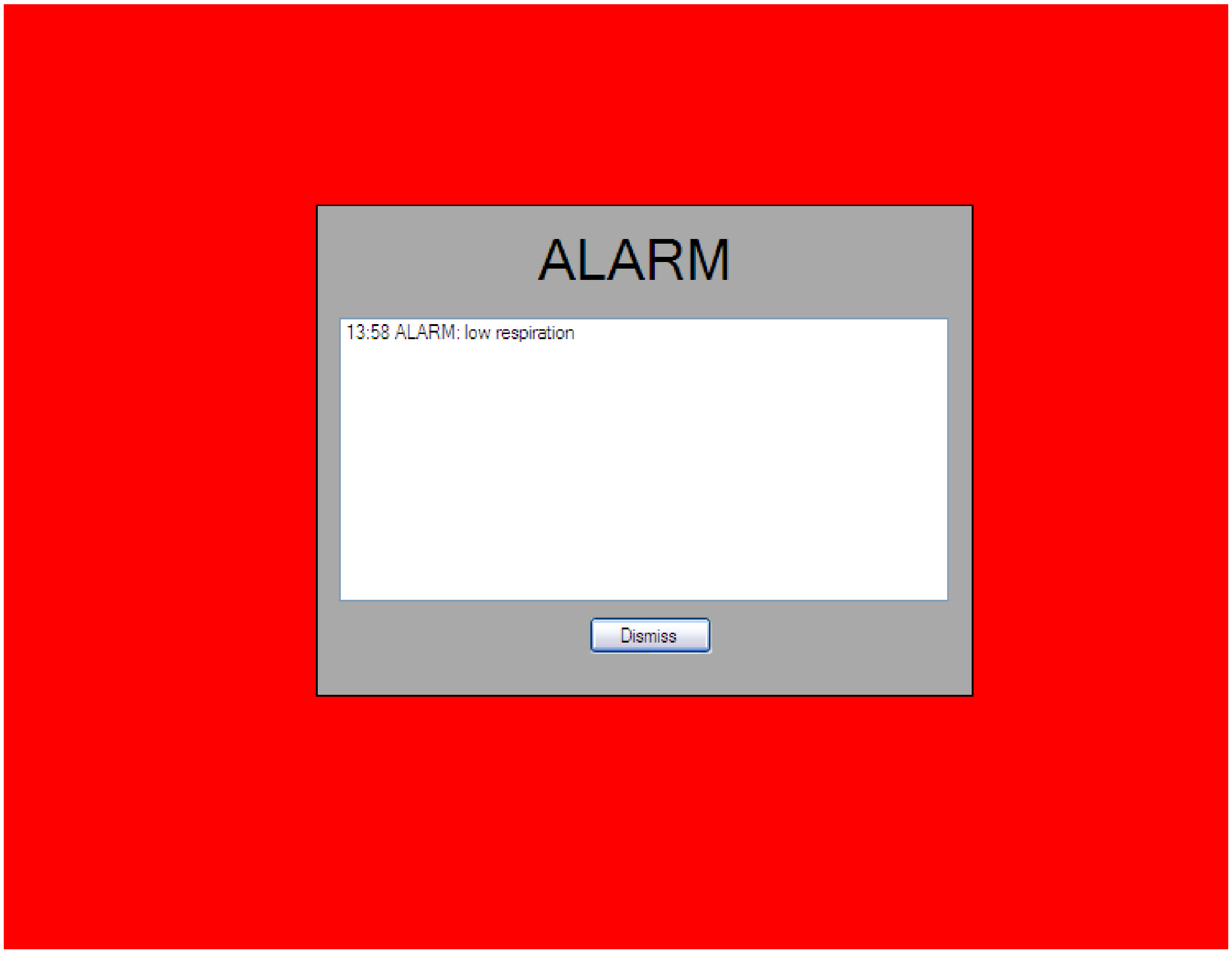
Threshold alarm high: 35
Threshold warn high: 26
BR: 0
Threshold warn low: 0
Threshold alarm low: 0

SPO2: 97%
Threshold warn low: 93
Threshold alarm low: 88

Motion: 44
Posture: 150
Temp: 36,7C
MedPacket id: 33780
Timestamp: 30.04.2008 13:53:57

Patient: Ola Nordmann
Monitoring ON

FIGURE 7-1 - MAIN WINDOW

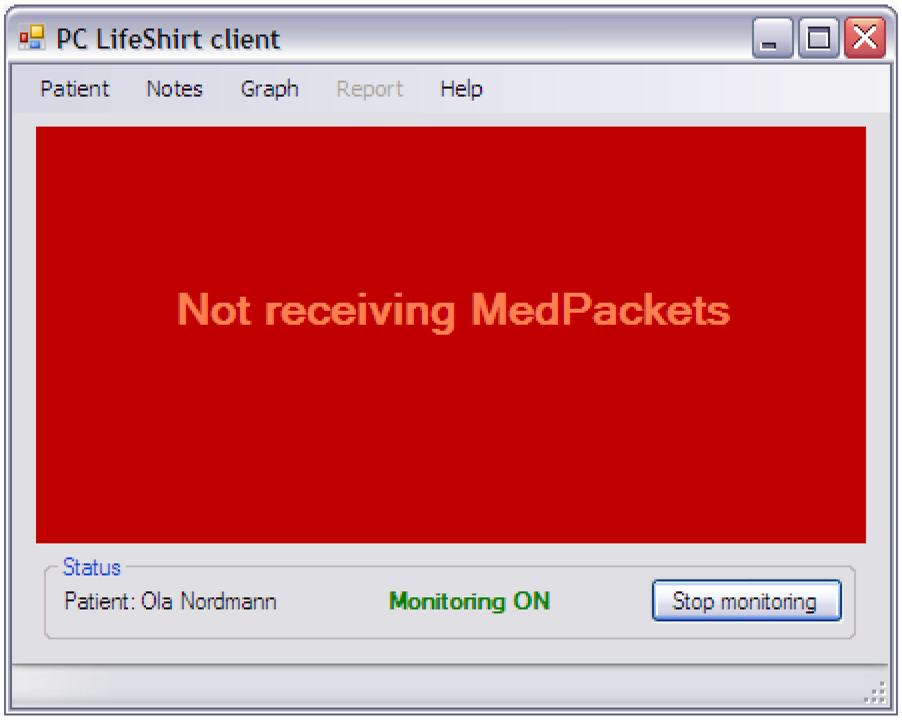


13:58 ALARM: low respiration

FIGURE 7-2 - ALARM WINDOW

Exceptions are thrown from the server to the client where the client halts, displays the error to the user and doesn't continue until the user has selected an action on the exception.

The PC client has exception handling for when it does not receive medpackets. If the client, for a period of 30 seconds, does not receive any medpackets, it will generate a "no-connection" alarm that is displayed in the normal fashion to the user. If the situation persists after the alarm has been dismissed, the client goes into no-connection mode until it once again receives medpackets, see Figure 7-3. Note that if the client doesn't receive a single medpacket it will ignore and continue; a "no connection" is only generated when packets are missed for 30 seconds.



Not receiving MedPackets

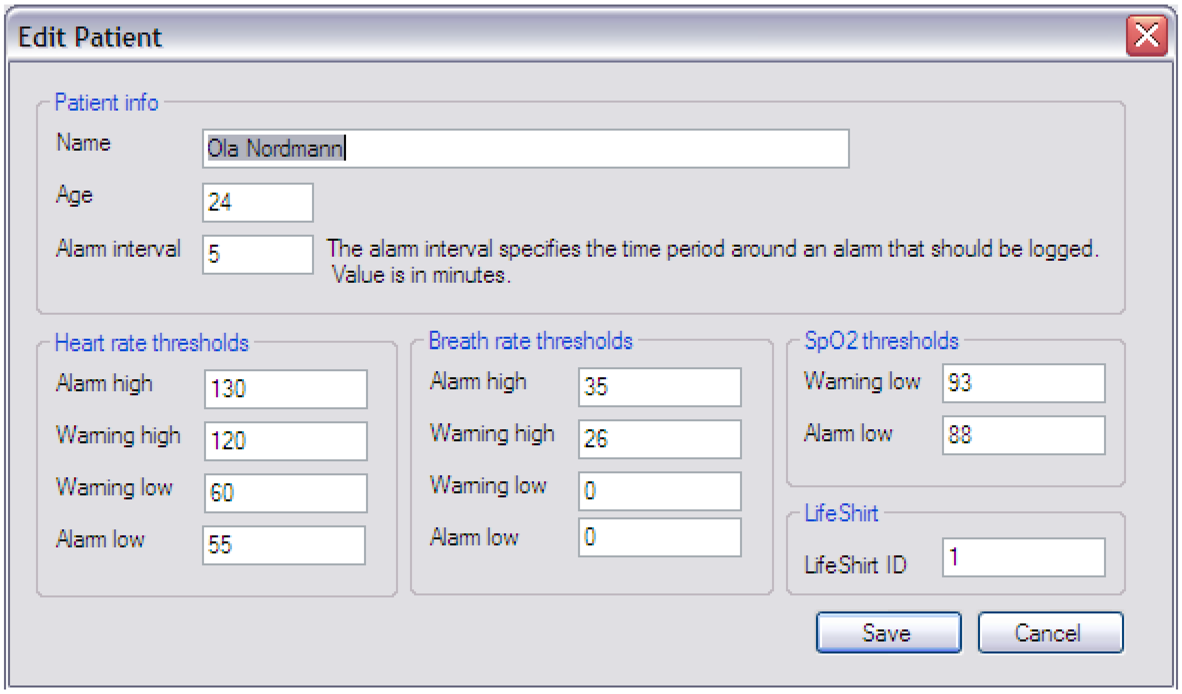
Status

Patient: Ola Nordmann

Monitoring ON

FIGURE 7-3 - EXCEPTION HANDLING

The PC client was then extended with rich functionality, so that patients may be added and edited and notes may be written; see Figure 7-4. Graph functionality as shown in Figure 7-5 were added as well, allowing a user to see stored medpackets.



Patient info

Name	Ola Nordmann
Age	24
Alarm interval	5

The alarm interval specifies the time period around an alarm that should be logged. Value is in minutes.

Heart rate thresholds

Alarm high	130
Warning high	120
Warning low	60
Alarm low	55

Breath rate thresholds

Alarm high	35
Warning high	26
Warning low	0
Alarm low	0

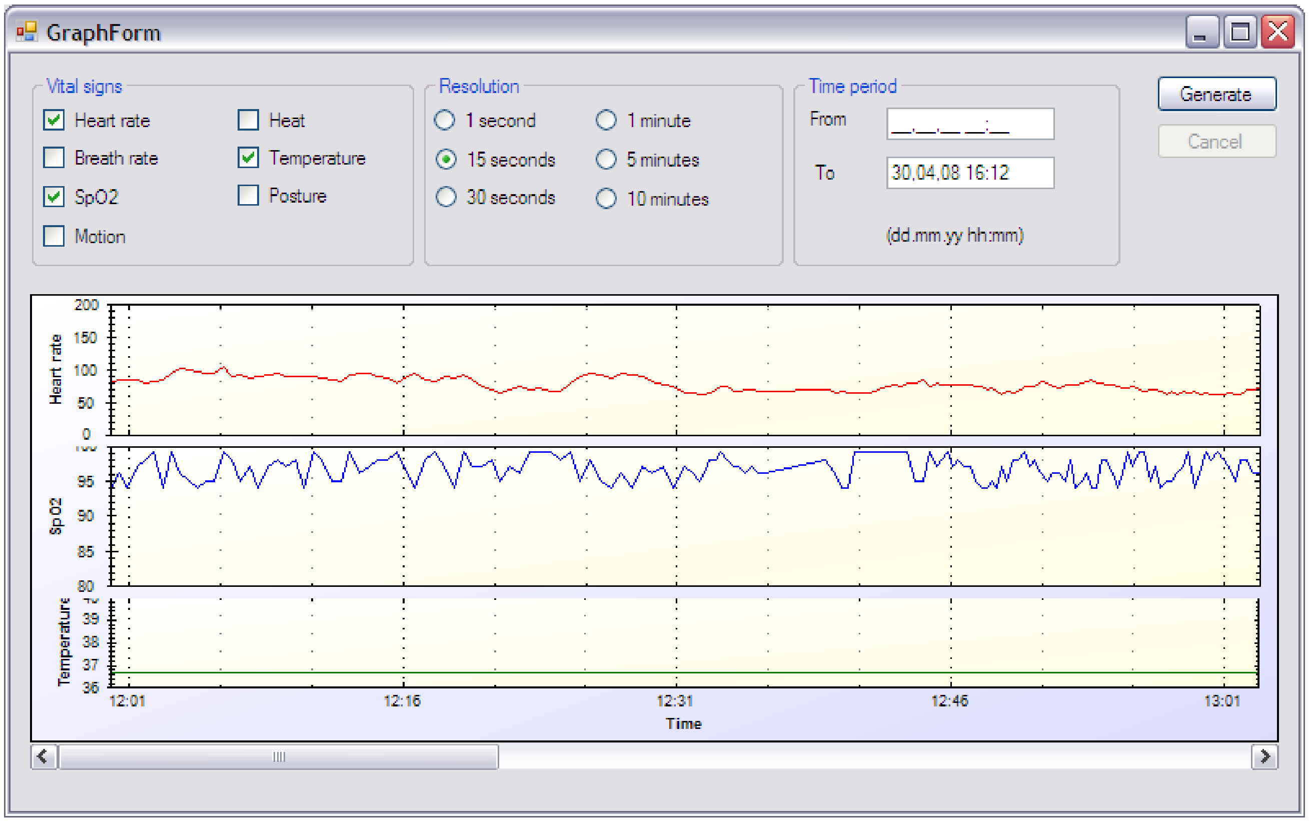
SpO2 thresholds

Warning low	93
Alarm low	88

LifeShirt

LifeShirt ID	1

FIGURE 7-4 - EDIT PATIENT



Vital signs: [✓ Heart rate] Heat, Breath rate [✓ Temperature], [✓ SpO2] Posture, Motion

Resolution: 1 second, 1 minute, [✓ 15 seconds], 5 minutes, 30 seconds, 10 minutes

Time period: From __.__.__ __:__ To 30.04.08 16:12 (dd.mm.yy hh:mm)

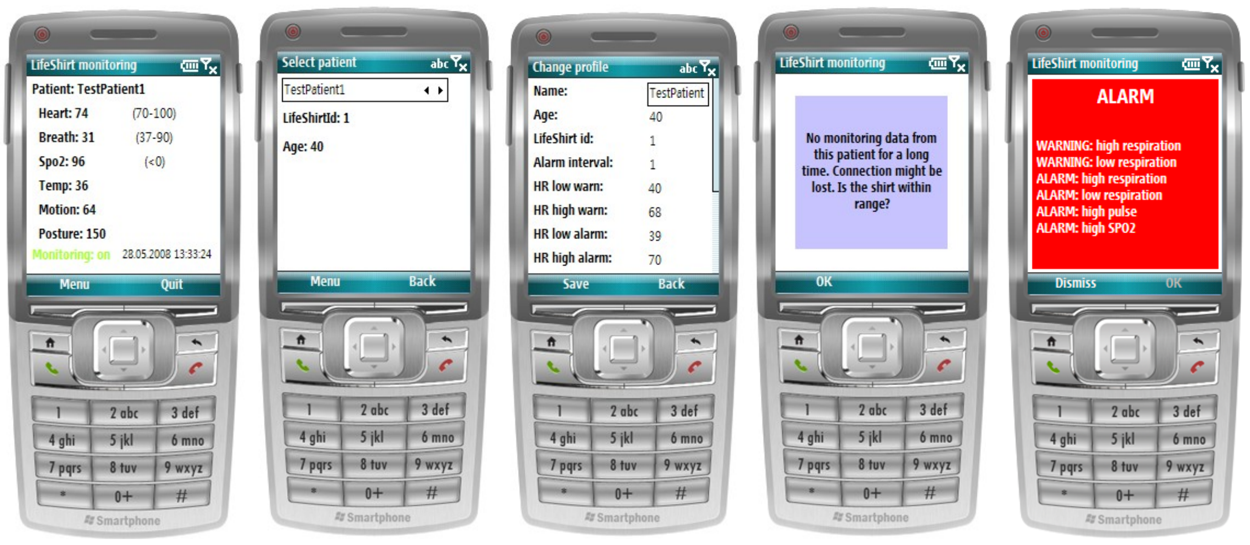
FIGURE 7-5 - GRAPHS

7.4 MOBILE CLIENT

It took some time to set up the mobile development environment. The mobile client requires internet, and getting the mobile emulator online was not a straight forward process. This required some extra software and set up. When this was done, it was crucial to get the web service to work with the mobile device. This was straight forward and worked in the same way as for the PC client.

The logic classes from the PC client could be used at the mobile client as well, and a simple form calling one web service method to get data and one to save data was created. This showed that the data flow worked from mobile client to server.

With the data flow working, the different forms for the mobile application were created and the layout was outlined. Figure 7-6 shows all screens available.



Patient: TestPatient1
Heart: 74 (70-100)
Breath: 31 (37-90)
Spo2: 96 (<0)
Temp: 36
Motion: 64
Posture: 150
Monitoring: on 28.05.2008 13:33:24

TestPatient1
LifeShirtId: 1
Age: 40

Name:	TestPatient
Age:	40
LifeShirt id:	1
Alarm interval:	1
HR low warn:	40
HR high warn:	68
HR low alarm:	39
HR high alarm:	70

No monitoring data from this patient for a long time. Connection might be lost. Is the shirt within range?

ALARM
WARNING: high respiration
WARNING: low respiration
ALARM: high respiration
ALARM: low respiration
ALARM: high pulse
ALARM: high SPO2

FIGURE 7-6 - MOBILE CLIENT FORMS

The GUI was made easy and clean, with the use of only panels, labels, textboxes and menu items. The only real time changes are in the color of the text and background. A simple layout increases the chance that the application looks as it should on several different mobile devices.

The left button on the menu was chosen to represent choices that take you further, or as an accept button to the user. The right button on the menu was chosen to represents a regret option where you can go back or quit the application. An exception of this is the alarm-form where there is no option to go back.

For the alarm form, red was chosen to represent an alarm, and yellow to represent a warning. These colors attract attention which is important. The message box has a light blue background color. This is to show that there is a message but not a life threatening one such as the red color for alarms. If another client than you dismisses an alarm, you get a message telling this. The alarm form will continue to show until you dismiss the alarm (or someone else does it). This means that an alarm may be visual even though the patient's vital signs are back to normal.

An alarm also has a sound connected to it, and functionality for vibration start and stop. The sound is an alarm sound which plays over and over again until stopped when the alarm is dismissed. We had some problems embedding the sound into the application, but the sound should now work on any Windows mobile 5 or 6.

The vibration works fine on the emulator but has unfortunately not worked on our test phone. The API says that it should work on any smartphone newer than 2002, but we have not used too much time troubleshooting this issue. Due to our time limitations this has been considered less important.

As mentioned the sound doesn't stop playing until the alarm is dismissed. If the application is shut down using the red phone button located on many phones (which terminates whatever you are doing with your phone) the sound will continue playing. This is obviously not an ideal solution and should be fixed in future iterations.

Exception handling is also present in the mobile application. All exceptions are caught in the main form and message forms are created with an appropriate message. This masks all errors and the user are informed that something went wrong and encouraged to try again.

When an error occurs, the monitoring is automatically stopped. The user can see this as a red text "monitoring: off" in the main form. The user can also manually start and stop the monitoring whenever he/she likes. It is always indicated in the main form whether the monitoring is on or off.

7.5 SEVERAL CLIENTS AT THE SAME TIME AND USE OF THE LIFESHIRT

With the implementation of clients and server finished, it was time to use it all together as a whole system; the actual LifeShirt, server, PC client and mobile client. During the implementation we used the simulation software provided by VivoMetrics.

For testing during implementation and during the test phase, we configured the LifeShirt and the mobile client to connect to the same network. This network used a static configuration and was hardcoded; a static ip was set for the LifeShirt and the mobile client as well as the ip where the server was located.

This was however not an optimal solution, since both the LifeShirt and the mobile client was limited by the access point's range, thus reducing mobility. The LifeShirt supports dynamic configuration which should be used in a real situation.

We have had some issues with the LifeShirt and non-standard network solutions. When encrypting the test network, the LifeShirt would freeze and be unable to connect. Installations from VivoMetrics are usually in controlled environments which need only a static configuration; consequently LifeShirt functionality for dynamic networks is not thoroughly tested.

For our test phase the static configuration worked sufficiently, however for use in a real situation the network setup should be modified to the surroundings. This is further discussed in the chapter 9 - Summary and discussion.

7.6 REMAINING FUNCTIONALITY

Due to time limitations we were not able to implement every requirement and this section briefly describes the remaining functionality not yet implemented.

Support for report generation was not implemented. The idea is that medical personnel could use reports based on monitoring data to better understand the patient's behavior and vital signs. Data should be searchable based on different input, like dates, alarms, vital signs and so on. All necessary data are already stored and the remaining part is to get the correct data and present it in a readable and printable way. Also, the possibility to change the frequency of logging is missing. Now medpackets are saved each second whenever monitoring is on.

Help functionality is also missing in the application. This is meant as guidance for the user and a description of for instance normal range in vital signs. Help functionality is important in a system with different users, but with our specific case this was not important to finish at this point.

The mobile client lacks the opportunity to write notes and save them together with a timestamp. This should be easy to add at a later point.

Table 7-1 contains a list of the functional requirements missing in the system.

ID	Description	Comment
FR 2.4	Data should be presented on the figure of a person. Vital signs readings should be placed on the anatomically correct part of the figure's body, e.g. the heart rate close to the heart of the figure.	Not implemented due to its low importance
FR 2.7	Monitoring should be possible to perform in different WLANs	Not core functionality for our scope
FR 2.8	Vital signs captured should be stored in a central database	Not important with only one user
FR 3.7	The alarm should give a audible clue to what triggered the alarm	Not implemented due to its low importance
FR 4.1-FR 4.3	PC application should have in depth explanations of vital signs and step-by-step guides	Not implemented due to its low importance
FR 5.1-FR 5.7	Report generation based on options chosen by the user.	Not implemented due to time constraints.
FR 6.1-FR 6.2	Different frequency of monitoring data logging	Assuming infinite storing capacity for now.
FR 7.2	In the mobile client it should be possible to make notes. These notes are stores together with their time of creation	Not implemented due to time constraints.

TABLE 7-1 – REMAINING FUNCTIONAL REQUIREMENTS

Table 7-2 contains a list of the quality requirements missing in the system.

ID	Description	Comment
QR 4	Help functionality, end user doesn't understand what sensors mean and looks up in help function	Not implemented due to its low importance

TABLE 7-2 - REMAINING QUALITY REQUIREMENTS

8 TESTING AND EVALUATION

"We must turn to nature itself,
to the observations of the body
in health and in disease
to learn the truth."
- *Hippocrates*

This chapter describes the testing phase and the results from the testing. It starts with a short introduction to *technology acceptance model* (*TAM),* which we loosely used as the model for the practical testing with the specific user case. Then some test cases are presented, which gives an introduction to the system and its functionality. The results from the first testing are then described and concluded based on TAM. Thereafter test results from the second testing period are presented. The chapter ends with a discussion around the reliability of the test results.

8.1 TECHNOLOGY ACCEPTANCE MODEL THEORY

TAM is a theory that models how users come to accept and use computer information systems. It was developed by Bagozzi, R. P., Davis, F (Bagozzi & Davis, 1992) in 1992. Based on empirical studies they modeled how different factors influence the use of a technology or a system. Most notably are the *perceived usefulness* and the *perceived ease of use,* together these lead to *intention to use*.

The usefulness of a system was defined by Davis as: "*the degree to which a person believes that using a particular system would enhance his or her job performance*" and the ease of use as "*the degree to which a person believes that using a particular system would be free from effort*".

These factors highly influence later use of a system, so the stakeholders' views on these factors were very interesting.

8.2 TEST CASES

The test persons were the involved parts in Linda's life. That is, her family and medical personnel. To make persons involved in the test familiar with the system, we modeled it as a set of test cases. These cover all important implemented functionality. These test cases then represented a basis for acceptance discussion and evaluation.

In the walkthrough of these test cases a few issues were discovered. These are described in the textual test cases below.

Figure 8-1 presents the test cases for the mobile user.

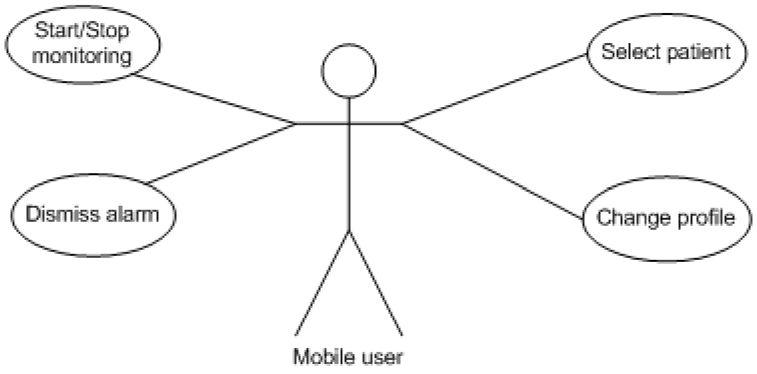


FIGURE 8-1 - MOBILE USER TEST CASES

Figure 8-2 presents the test cases for the PC client user.

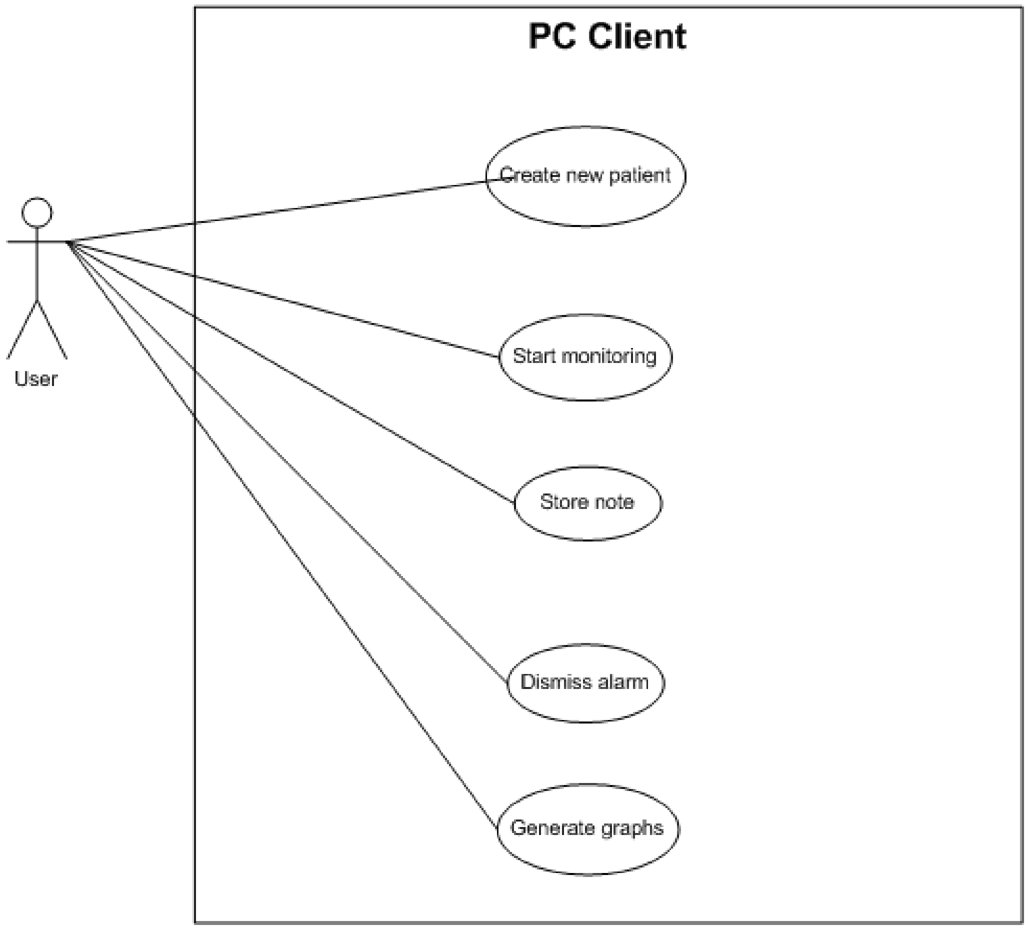


FIGURE 8-2 - PC USER TEST CASES

8.2.1 TEXTUAL TEST CASE TEMPLATE

Test Case name	
ID	
Description	
Accompanying requirements	
Expected course of events	
Alternative paths	
Exception paths	
Observed events	
Deviation analysis	

TABLE 8-1 - TEXTUAL TEST CASE TEMPLATE

The sections in this test case template should contain the following:
Test case name refers to the names from Figure 8-1 and Figure 8-2
ID is a unique number for each test case.
Description contains a brief description of the test case.
Accompanying requirements connects the specific requirements from chapter 4 to the individual test cases
Expected course of events contains a list with the expected sequence of actions in the specific test case.
Alternative paths represent the possibilities the user has to perform other choices and still accomplish the task, other than the one listed in the previous section.
Exception paths lists the possible exceptions that may occur when the user goes through the sequence listed in expected course of events.
Observed events describe events that are abnormal.
Deviation analysis analyzes differs in expected and observed course of events.

8.2.2 TEXTUAL TEST CASES

Below is a textual description of all the test cases from figure Figure 8-1 and Figure 8-2, one table for each test case. The PC client test cases are given id UC-X, where x is a number from 1 to 5, and the mobile client test cases are given id UCM-X, where x is a number from 1 to 4.

8.2.2.1 PC CLIENT TEST CASES

Test Case name	**Create new patient**
ID	UC 1
Description	The user creates a new patient with thresholds specific to the patient.
Accompanying requirements	FR 1.1, FR 1.4
Expected course of events	1. User selects "Create new patient" 2. User inputs patient specific information and information that deviates from standard values 3. User clicks "store patient" and patient is stored to the server
Alternative paths	-
Exception paths	2.1 User inputs illogical values (e.g. min heart rate of 200) 3.1 The user has entered not a number where the form requires a number. The error is presented to the user and the user may correct the error before trying again. 3.1 An error occurs in the communication with the server. The server may be down or a communication error may occur. The user is presented with the error message and may either try again or cancel the process.
Observed events	Nothing unexpected occurred

TABLE 8-2 - CREATE NEW PATIENT TEST CASE

Test Case name	**Start patient monitoring**
ID	UC 2
Description	The user selects a patient and chooses to start patient monitoring in real time.
Accompanying requirements	FR 2.1, FR 2.3
Expected course of events	1. The user clicks "Select patient" 2. The user selects the desired patient from a list of patients from the server 3. The user checks "Monitor patient" 4. The user clicks "ok" 5. Monitoring is started and data flows into the main window
Alternative paths	3.1 The user does not check "Monitor patient", active monitoring is not started
Exception paths	The server is down and does not return a list of patients 5.1 The client does not receive any medical packets from the server. After 30 seconds the client alerts the user that it is not receiving any packets from server.
Observed events	Nothing unexpected occurred

TABLE 8-3 - START PATIENT MONITORING TEST CASE

Test Case name	**Store note**
ID	UC 3
Description	The user is able to store a note about a patient. The note is stored in the server with timestamp and which patient it is associated with.
Accompanying requirements	FR 7.1
Expected course of events	1. The user has selected a patient earlier in the session. 2. The user selects “Make note”. 3. The user inputs note comment. 4. The user selects “Store” and the note is stored to the server.
Alternative paths	-
Exception paths	No patient selected, “Make note “ not available 4.1 No connection to server, or an error occurred when communicating with server. The user is presented with the error message and may retry or cancel the operation.
Observed events	Nothing unexpected occurred

TABLE 8-4 - STORE NOTE TEST CASE

Test Case name	**Dismiss alarm**
ID	UC 4
Description	The user is presented with an alarm screen. The alarm is audible and visible. The user can dismiss the alarm and input a text that is stored with the alarm.
Accompanying requirements	FR 3.1, FR 3.4, FR 3.5, FR 3.6, FR 3.8, FR 3.9
Expected course of events	1. Alarm monitoring is initialized earlier in the session. 2. The client receives an alarm from the server and displays it to the user. 3. The user, upon having reviewed the alarm, clicks “dismiss”. 4. A form is displayed to the user where the user can input text. 5. The user clicks store and the dismissed alarm is stored with the text.
Alternative paths	3. Another user has dismissed the alarm. This information is displayed to the user. The user clicks ok.
Exception paths	Unable to store the dismissed alarm to server due to communication or server issues. This information is displayed to the user. User must verify connection to LifeShirt is functioning.
Observed events	Nothing unexpected occurred

TABLE 8-5 - DISMISS ALARM TEST CASE

Test Case name	**Generate graphs**
ID	UC 5
Description	The user can create graphs of vital sign for a time period.
Accompanying requirements	FR 2.5, FR 2.6
Expected course of events	1. The user has selected a patient earlier in the session. 2. The user inputs which vital signs he is interested in and the time period. 3. Data is retrieved from server and displayed to the user. 4. The user may zoom and pan in the graphs.
Alternative paths	-
Exception paths	No user selected earlier, “generate graph” option not available. Server is down or communication problems. The error message is presented to the user. User may dismiss the error message and try again or cancel the operation.
Observed events	An error was reported. When trying to select the time period just monitored no data were presented.
Deviation analysis	Checking the monitoring data in the database showed an error in the time stamping of the medpackets. The date of the monitoring data were days earlier than the actual monitoring.

TABLE 8-6 - GENERATE GRAPHS TEST CASE

8.2.2.2 MOBILE CLIENT TEST CASES

Test Case name	**Select patient**
ID	UCM-1
Description	The user selects a patient to monitor from all available patients
Accompanying requirements	-
Expected course of events	1. The user chooses Menu->Patient->Select patient 2. A combobox with all available patients are presented at the screen 3. The user navigates between the patients using arrows 4. When the patient wanted are shown, the user chooses Menu->Choose 5. The patient are chosen and monitoring starts automatically
Alternative paths	4.1 The wanted patient is not in the list and the user chooses Back 4.2 The user sees the earlier patient, or no patient if no earlier patient is present
Exception paths	2.1 No patient to retrieve, the user gets a message that no patient is stored 2.2 No patient is retrieved because of an error at the server, the user gets a message about this 5.1 Unable to start monitoring, the user gets a message that no monitoring data is available
Observed events	Nothing unexpected occurred

TABLE 8-7 - SELECT PATIENT TEST CASE

Test Case name	**Change profile**
ID	UCM-2
Description	The user wishes to change the thresholds or other patient information, and saves the changes to the server.
Accompanying requirements	FR 1.3, FR 1.4
Expected course of events	1. The user has already selected a patient and chooses Menu->Patient ->Change profile 2. The profile is presented with all fields 3. The user changes one or several fields 4. The user chooses Save 5. The new patient profile is saved at the server side
Alternative paths	1.1 The patient selected is not the patient the user wants to change 1.2 The user chooses Menu->Patient->Select patient 1.3 The user navigates to the patient wanted 1.4 The user chooses Menu->Change profile 4.1 The user regrets changing the profile and chooses Back 4.2 The changes are not saved
Exception paths	4.1 The user has used letters as input to number fields 4.2 A message describing that only numbers from 0-9 should be used are shown 4.3 The profile are set back to the initial fields 5.1 Unable to save the new patient information at the server side 5.2 A message describing this is presented
Observed events	Nothing unexpected occurred

TABLE 8-8 - CHANGE PROFILE TEST CASE

Test Case name	**Start/stop monitoring**
ID	UCM-3
Description	The user can choose to stop the monitoring if it is running, and start the monitoring if a patient is chosen and the monitoring is stopped.
Accompanying requirements	FR 2.2, FR 2.3
Expected course of events	1. A patient is selected and the monitoring starts automatically 2. The user chooses Menu->Monitoring->Stop to stop the monitoring 3. The user chooses Menu->Monitoring->Start to start the monitoring
Alternative paths	No patient is selected The user selects a patient, see UCM-1
Exception paths	2.1 Unable to start monitoring, a message is shown which describes this
Observed events	Nothing unexpected occurred

TABLE 8-9 - START/STOP MONITORING TEST CASE

Test Case name	**Dismiss alarm**
ID	UCM-4
Description	When an alarm goes off, this is presented as a red screen, a sound and the mobile starts vibrating. The alarm has to be dismissed by a client, mobile or PC.
Accompanying requirements	FR 3.2, FR 3.4, FR 3.5, FR 3.6, FR 3.8, FR 3.9
Expected course of events	1. An alarm goes off 2. The user chooses Dismiss 3. A textbox where the user can write a note is presented 4. The user writes a note and chooses OK 5. The alarm is stopped and the monitoring screen is shown
Alternative paths	The alarm is dismissed by another client The user gets a message describing that the alarm was dismissed
Exception paths	4.1 Unable to save that the alarm was dismissed, a message describing this is shown
Observed events	A weakness was reported. When dismissing an alarm a new one came up instantly.
Deviation analysis	This happened because the vital signs might still be abnormal when an alarm is dismissed. In these cases new alarms will be generated instantly, and the user would have to dismiss alarms continuously.

TABLE 8-10 - DISMISS ALARM TEST CASE

8.3 RESULTS FROM THE FIRST TESTING SESSION

The first test session took place at a hospital in Norway. The involved parties were Linda's father and the chief physician at the pediatrics division. We started out with an explanation of the system and continued with a discussion. Then we did a functional test together with Linda's father. Due to time constraints, the physician was unable to attend our functional demonstration. However, he gave us an evaluation of the system in general, answering medical questions, which is further discussed in chapter 9 - Summary and discussion.

For the demonstration of the system Linda's father wore the LifeShirt, thus familiarizing himself with setup of the system. Together we went through all the functionality of the system, all the time discussing. This discussion proved valuable for both parties as we understood each other better.

8.3.1 WEAKNESSES AND POSSIBLE SOURCES OF ERROR

Some errors and weaknesses were discovered through the first test session. As described in the earlier textual test cases, the generation of graphs was unsuccessful. Upon an inspection of the database an error with the timestamp was discovered, thus the program did not find any medpackets in the given time interval. This bug was found and corrected.

A weakness in our solution that often can occur, is that right after an alarm is dismissed, another alarm is generated since the vital sign has yet to return to normal values. In discussion with Linda's father we evaluated whether this was wanted functionality or not. It was not. The monitoring solution they currently use will not generate a new alarm for a 60 second period after an alarm is dismissed. Therefore we decided to implement the same functionality in our system. More advanced solutions can be implemented; we discussed *hysteresis,* however that was not required at the current stage.

A system that uses *hysteresis* will have a number of states, and the reaction to input will depend on which state the system is currently in. In our case, there would be different logic depending on whether an alarm has just been sounded or not for a vital sign value outside of the norm. Our solution now use a simplified version of hysteresis by disabling alarms for 60 seconds after an alarm is dismissed, thus our system has two states: *nothing special* and *alarm just dismissed, do nothing.*

We have had some issues with the SpO_2 reading, when the sensor is unable to properly read a value, it will send -1. Our system will ignore one failed reading; however ten unsuccessful readings in a row will trigger an alarm. Linda's father thought it unnecessary to do a more complex alarm triggering, like for instance connecting the SpO_2 reading with the motion reading, in order to keep the system simple.

8.3.2 EASE OF USE AND USEFULNESS

The perceived ease of use of our system was twofold. First time setup and configuration is difficult and made for technology minded people, a layman that is unsure of technology would not be able to configure the system by himself. Once the system is properly configured, the perceived ease of use is distinctly higher.

The initial perceived usefulness of the system was high, not surprising since our system has been made based on Linda's father's functional requirements. The stakeholders were eager to try out the system at home in actual use straight away, so the system was definitely perceived as useful. Further testing with actual use gave better input from the stakeholders on the usefulness as well as the ease of use.

8.4 RESULTS FROM THE SECOND TESTING SESSION

As long as errors found in the first test session were corrected, Linda's father wanted to try the system at their home as soon as possible. All errors and requested improvements were fixed.

The system was then installed and configured (as much as possible from a remote location) on a laptop and sent to Linda together with the LifeShirt. Linda's father was guided through the rest of the installation process and he was able to make the system work without any major problems.

Medpackets, in a specified interval before and after an alarm, should be attached to the alarm in the database. This requires a lot of database operations and since there is no possibility to fetch monitoring data based on alarms in the developed example clients at this point, this was removed to release resources.

Linda and her family performed testing at their own leisure in their home, after which we received feedback on the solution.

8.4.1 WEAKNESSES AND POSSIBLE SOURCES OF ERROR

There were discovered several weaknesses during testing. The most important was the limited battery capacity of the LifeShirt. One evening Linda was set up for monitoring at 19:00, at 02:00 the batteries were empty. This was not sufficient since monitoring was planned for the whole night.

The size of the LifeShirt proved to be a major issue. The family thought there were too many wires and that the PDA was too big compared to Linda, so that the solution was uncomfortable to wear for her.

In addition they thought the LifeShirt was too much for their needs, which was to monitor the heart rate and SpO_2. They would rather have a simpler solution, more resembling their current one. They wanted especially to replace the laptop with something simpler, for instance a black box with an on/off switch and some lights to indicate status.

When the connection is lost on the mobile clients an alarm is sounded, however this alarm was not audible enough. This bug was found and corrected. Another feature request was to see the value of the vital sign that triggered an alarm.

The users found the concept promising, but that the system requires improvements in usability, energy consumption and clinical trials to ensure user confidence.

8.4.2 EASE OF USE AND USEFULNESS

There were some major issues found during the test session that greatly impacted the perceived ease of use and usefulness.

The users found the system to be complex and wanted something simpler, more resembling their current system described in section 2.3.2. The laptop and the startup of the system were too complex, which negatively influenced the ease of use.

Due to the battery issue and the alarm that was not audible enough, the users lost confidence in the system, and consequently the perceived ease of use and usefulness dropped.

The perceived ease of use and usefulness were not high enough to warrant further testing with the system in its current form. However the users acknowledged the usefulness of the concept, and that with improvements it would be something they would want to use in the future.

8.5 RELIABILITY OF RESULTS

Results and evaluation from testing of the system were based on testing with the specific user case. The results from the system developed were mainly positive and it was the LifeShirt which caused trouble when taking the system in use. The system was developed based on requirements from the specific case and therefore it made little sense to test it on other potential users. It is important to have in mind that although this family was positive to the system as long as the shirt was made more comfortable, another family might have completely different feedback.

Linda's father is above average interested in technology and has no hesitation with new and unfamiliar systems. Linda's doctors have no clue on what causes her seizures and that is very frustrating for all people involved. Therefore, anything that might help with the situation is very welcome. This might not be the case for other potential users, and could be a weakness in the test results.

Linda's father has studied and worked with technology for a long time, and his knowledge of computers is not comparable to a layman. This means that although he was able to set up the system at his home, this might not be the case for other users. Also, he automatically understood more of the system and what was behind, which probably made it a lot easier to use. He also knew what to expect and had a realistically view of the system. This could have made the results less reliable because Linda's father's threshold for acceptance was quite low, which might not be the case for other users.

On the other hand, Linda's father's knowledge made the feedback more detailed and weaknesses of the system that a layman probably wouldn't have discovered could now be reported. He could also give feedback on more than just the system's appearance. This made the results more reliable.

The fact that we were able to test the system in a real environment made the results more reliable. The system was set up in the patient's home and used for a night. Earlier tests of monitoring systems have been performed in demonstration environments. Although this gives useful information as well, testing in a real environment gives another perspective of the system.

9 SUMMARY AND DISCUSSION

"To keep the body in good health is a duty...
otherwise we shall not be able to keep our mind strong and clear."
- *Buddha*

This chapter gives a brief description of our contributions, and goes on to evaluate and discuss different aspects of the system and the test results.

9.1 CONTRIBUTIONS

We have made a system with functionality not found on the market today; i.e. alarm functionality on cell phones. Our system has been tested in a real situation with a real patient; it is not just a prototype in a lab setting. The views we have received from the family of the patient and the physicians are valuable for further research in this field.

9.2 TECHNICAL ASPECTS OF OUR SYSTEM

The following section provides discussion and evaluation of the different aspects of our system.

9.2.1 INSTALLATION AND USE

Since we haven't made a production version, installation of the system is difficult. As mentioned in section 1.2, this was outside our scope. All server components and client developing environments have to be installed on the computer running the server. Once the server is running it may run without user interaction. Then the usage of the mobile and the PC client are easy.

When the server is up and running, the web reference on the mobile client has to be updated before it is compiled. This is due to the nature of web services. Usually a web service would be looked up through a DNS hostname, not a static ip, and thus it would not have this problem.

Our mobile development version is just transferred to the mobile, it isn't installed like a normal program, and thus it doesn't show up in the program list, you have to manually start it from the file system explorer. A future version should integrate better with the operating system.

The exception handling in the software should ensure that the system will function even though an exception occurs. However if something happens, the error message is quite technical, e.g. a SoapException indicates something wrong occurred in the server-client connection. A layman would not feel comfortable with this detail level. A production system should mask technical information; however for the current use where the stakeholders are technical oriented this can be seen as a strength of the system.

Should a grave error occur that is unanticipated and not handled by the system, the whole system would have to be restarted. This is not user friendly, and a future version should be able to automatically restart system components when a crash occurs.

9.2.2 DIFFERENT LAYERS OF THE SYSTEM

We use different programming environments in the different layers of our system. This adds complexity to the system, and adds to possible error sources. On the other hand, the use of layers makes the system easier modifiable and easier to understand for new developers.

In order to connect to the LifeShirt data we had to use several systems, an evaluation of this connection is detailed below.

9.2.2.1 SERVER LAYER

Connecting to the VivoMonitor server was error prone; the programming environment used for this connection is considered legacy Windows. Another weakness in this link is that all data becomes stored in the VivoMonitor server component as well. We do not know how this server will scale with an ever increasing amount of data. Periodic erases of the server database may be done, but this is a suboptimal solution.

Our server layer runs as two processes, one reads and stores medpackets while the other process runs on an application server and exposes the web services. Our first attempt was to integrate the MsgClient functionality (described in section 7.1) into our server. However due to the complexity of this integration and time constraints we were unable to get it to work, though it should be done in a future system. Should this integration be possible, with the Java process receiving the medpackets, the server system would be cross platform, which could be a great advantage. At the moment the server has to run on Windows, which is a weakness.

To run our server we have to run 4 different processes (not counting the database processes): VivoMonitor server, MsgClient, Java read-in and processing, and Java WebService server. Each process adds complexity and possible error sources, so future versions should try to limit the number of processes. Using a file to communicate between different systems is not elegant; however it has proved stable during development and testing.

9.2.2.2 CLIENT LAYER

Our goal was to use a model-view-controller architecture (MVC), so that as little logic as possible would be in the clients. However, due to the current web service standard we had to put some logic in the clients that should instead be in the server layer. At the moment it isn't possible to use an event and listener model with web services. With this model the clients could just react to events that the server generates, for instance the server would emit an alarm-event to all clients, and the let the clients display the alarms themselves. In our system the clients will actively ask the server to see if there is a new alarm.

Since the clients actively asks for new medpackets and alarms all the time, checking the state of the connection to the server is done automatically. However the clients require logic for deciding whether the medpacket or alarm that it has received is something new, and then what to do with the alarms. Had the server been able to alert the clients that for instance a warning has been upgraded to an alarm, the client logic could become even simpler, and more resembling a MVC architecture.

We have programmed the clients in current software languages that should be relevant for many years to come. At the same time we have limited the number of fancy features and eye-candy in order to keep everything simple so our clients won't require certain special system requirements. Our clients may be extended with whatever functionality is required for other use cases than the one we have developed for.

At the moment the PC-client is run on the same computer as the server, however, this is not necessary. Separating the PC-client and the server is done very quickly. The server could then be stowed away in a server warehouse with all the functionality remaining the same.

While our clients have some weaknesses, they are made with robustness in mind and should function properly over time.

9.2.3 TECHNOLOGIES USED

We opted for the .Net environment on the clients for a number of reasons. .Net has great support for creating graphical user interfaces, which was nice due to our time constraints. Since the server has to run on Windows, it was not necessary for the PC-client to be cross platform. .Net is an environment that is easy to develop for, with a managed memory environment, as well as the web service technology fully integrated. For the mobile client using the .Net environment was required since both the mother and father of Linda have Windows Mobile. Using .Net on both clients allowed code sharing between the two clients.

The two mobiles which Linda's parents were going to use run different versions of the Windows operating system (version 5 and 6). There have been numerous changes between these versions and we initially thought it would be difficult to program a client that would work on both. With this in mind we limited the use of fancy features, concentrating on core functionality, so the mobile client would be compatible with a wide variety of devices. We succeeded and our client functions perfectly on both versions and on the two different devices we have had available for testing.

The use of web services adds complexity and restrictions. The server component has to be run on Java 1.5 or higher. And using web services between Java and the .Net clients is not straight forward. We had to use the *RPC/encoding style* to get it to work. This is a style that is not WS-I compliant. WS-I is an organization that was formed to establish best practices for web service standards (Inc, 2008). The fact that *RPC/encoded* is not approved from WS-I indicates that this style could be removed from web service engines. This is a weakness to the system and should be considered to change in later versions.

The development of web services, both the exposing and consuming, was cumbersome; however once we got it to work it has proved very stable and flexible. Our server is service oriented, and existing clients and new clients may use just the services they require. This approach and technology makes it easy to extend and adapt clients as well as create new clients in the .Net environment or other programming platforms.

Windows mobile is an exciting platform which is easy to develop for. At the moment the platform is only available on high end smartphones, a mobile client written in Java mobile edition would have run on a larger set of the mobiles out there today. No matter which technology is the best for mobile programs, our mobile client may be converted to other languages relatively easy.

Use of Hibernate provides database abstraction, however Hibernate itself may produce errors. We had some problems during development with the same Hibernate object on multiple clients, when a change to the object was done on a client; the object on the other client was invalidated. This is the way Hibernate works; however this produced unpredictable errors at a later time, which was hard to debug. Similar problems could occur in the future rendering the database inconsistent so the server crashes.

We also had problems with Hibernate performance. Hibernate produced very time consuming SQL queries for what we thought was easy operations. This is because we do not know Hibernate that well, so our approach resulted in inefficient Hibernate use, especially when the database size grew large.

9.2.4 LOGGING FREQUENCY

A wish from the stakeholders is the logging of vital signs at different frequencies, but we are already logging at the max frequency that we have access to (1 Hz), therefore we have not done anything regarding this. Monitoring over longer periods will generate a lot of data for uninteresting stretches of time which is why we implemented the storage of medpackets in a time period around an alarm. The thought was that all medpackets not connected to an alarm would be deleted periodically. We do link medpackets to alarms, but we have not made any policies on when or how the rest of the medpackets should be deleted.

We have previously mentioned the use of hysteresis in order to have multiple system states. With different states the logging may be better at the important time periods, while ignoring the uninteresting. The aim was that when a certain threshold was breached, the system would go into a high-frequency-monitoring state and it would log everything that happens in detail. This functionality shouldn't be too hard to implement, however it requires a different API than the one we had access to.

9.2.5 ROBUSTNESS

Everything has been tested throughout development. While everything runs stable, the system hasn't gone through rigorous testing to enable it for use in life or death situations. Since our solution has many layers and use a varied set of technologies to communicate internally, there are a lot of possible error sources.

Should our system be disrupted and become unable to function the user would be alerted, but the user will most often not understand what the reason for the error is, and what he can do to correct the error. The user basically can just restart the system and hope that it will function again. A future system should be able to debug without being a developer.

9.2.6 MODIFIABILITY

One of the important aspects when developing this system was that it should be easy to modify to use in other use cases and for other patients than our specific user case.

Our focus has been on making the server component modifiable, the clients we have made are more for example use and may be more difficult to adapt and extend for future developers. Our server component has the flexibility of being able to change at both ends, both in how it receives medpackets and what it exposes as a server to clients.

The server oriented approach allows client to implement only the features they want in their client, so that the clients may be adapted to the hardware they run on.

Inside our server the degree of difficulty for modifying varies. The system can be easily modified to actively monitor for thresholds on other vital signs. More complex monitoring with trends and connecting vital signs monitoring together (such as if there's high motion ignore bad SpO_2 readings) is more difficult.

9.2.7 SCALABILITY

The main limit to scaling the system to multiple users is in the file output/reading which occurs between our server and the MsgClient system. If our server can receive the data in another way it may easily support multiple users and scale quite well. One server may provide services for multiple clients at multiple locations.

9.2.8 SECURITY

We haven't focused on security in our solution, since this wasn't in our scope. Securing the LifeShirt on the Internet as well as securing the communication between LifeShirt and server is something VivoMetrics will have to do. The communication between our server and our clients can easily be extended with login functionality and encryption of the communication between server and clients.

Security is an important issue if the system is to be extended in use and functionality. Monitoring data is very personal, and users need to know that this data cannot be misused. Since life can be at stake it is important that all parts and links of the system is protected against intruders.

9.3 THE LIFESHIRT TECHNOLOGY

The LifeShirt technology is very promising, but it has some weak points. In size and form factor the LifeShirt is both elegant and bulky. The main problem is the PDA, which is heavy and unpractical. The LifeShirt itself is slim and can be worn unobtrusively during normal activities. There is a great deal of wires required, some better hidden than others. When extending the core system with additional sensors there are a lot of extra wires involved. Exposed wires such as to the SpO_2 reader will be a problem when the monitoring is of active children.

If the PDA on the full size LifeShirt could be trimmed down and made smaller the LifeShirt would instantly become easier to wear. Notably eliminating the need to wear a pouch to carry the PDA in. The technology used in the PDA is some years old; today's PDAs have a smaller form factor. So the development of PDA technology works in VivoMetrics favor, the next model of the LifeShirt will probably feature a smaller PDA.

For our system we do not require the PDA as an input and output device at all, we only need to know that the LifeShirt is up and running. In this regard the VivoResponder, described in section 2.2, would be better for our use. It is more focused on just transmitting the data, so it does not have an input and output device like the PDA. A mix of the two models, providing all the sensors of the full size version together with the small transmitter of the lightweight model would be nice. A wider selection of LifeShirt models with different functionality would make it easier to customize a shirt for each patient's special needs.

Our system could be integrated with other monitoring shirts or equipment. However in order to gain approval for the use of a system beyond prototype trials, every part of the system would have to be approved by the authorities. Since VivoMetrics is already established in the market with approval from authorities, in both USA and Europe, they may easily extend their system with new sensors and functionality. New versions of the LifeShirt would also be easier to get approved than if a newly started company should start producing monitoring shirts.

The LifeShirt itself uses internet technology, but it is not made to operate on the Internet because of the strict security it then would have to implement. In a distributed scenario, where there's one server for multiple LifeShirt at different locations, the LifeShirt would have to connect directly to the Internet with all the security risks that implies. Before the LifeShirt could be used in this fashion, VivoMetrics would have to upgrade the software with a focus on security, and get it approved by relevant authorities.

Our solution targets another audience than the software shipped with the LifeShirt. VivoLogic and VivoMonitor are very advanced pieces of software with many possibilities. However for the home use we have developed for, it was a wish that the software would be simple and easy to understand for a layman. In our solution everything is still recorded on the memory card in the LifeShirt, so VivoLogic may be used for post-analysis. A future system would greatly benefit from closer integration with the LifeShirt and VivoMetrics' product line, so that laymen and physicians would get different views on the same data.

Further development depends on good cooperation with VivoMetrics, a tighter integration with the LifeShirt system, especially more access to the underlying software and a larger API. This would greatly benefit our system or a future similar system. VivoMetrics is cooperating with many research institutions, and they are very positive and eager to try out new usage areas for the LifeShirt.

9.4 LINDA'S FAMILY

This section evaluates how the system was in use and how Linda's family could be affected by it.

Linda's family could benefit from a monitoring system like the one developed in this thesis in many ways. Having a reliable monitoring system installed and in use gives the family more freedom. When Linda is sleeping, her parents can move around the house and even take a short trip outside the house. They will be notified immediately if something is wrong. Their limitations will then be how far they can move and still be able to get back in time, and not how far the baby monitor is working which is the case today. Being able to move inside the house is also a distinct improvement from today; now they have to look at the monitoring screen constantly.

Not only would Linda's parents be free to move around, Linda herself could walk freely as well. Today she is connected to a monitor at night, and at daytime she needs constant monitoring by another human being. With a personal monitoring system she could be anywhere inside the house as long as she is inside the wireless router's range.

Letting siblings monitor Linda would not result in the same responsibility for them. For siblings to know that their parents would be notified instantly would probably make it less scary. Also, it would be easier for Linda's parents to let other take care of her.

Today Linda's parents need to look at the monitoring screen at night to see that the monitoring is up and running. This would not be necessary with our system. If the monitoring stops working they would be notified instantly, as for the alarms. Also, they could start the monitoring on two mobile devices (both her parents has Windows mobiles), and therefore have double security if one of the mobile devices is accidentally turned off. This should give Linda's parents a good night's sleep.

In addition to Linda's parents, her doctors are involved as well. The monitoring data could be of importance to them. They can use it to raise discussions among doctors with different kinds of special fields, and see connections between different vital signs in her body. Also, having the monitoring data on paper in black and white would be comforting for Linda's parents, as they then know that the doctors aren't mistaken.

All these factors mentioned just now leads to an improved *health-related quality of life* which was discussed in section 2.4. Their everyday life would be easier and Linda's freedom as well as her parents' would increase. It is not possible to set a price on such an improvement, but it is no doubt that it is worth a lot to this family.

On the other hand, there are some points to be aware of. It is important to know the system and its limitations. It is not a treating system and it is not guaranteed that you will be able to diagnose with it. A danger would be to rely too much on the system and believe that it could perform things it can't. It is important that Linda's parents know every detail of the system and don't get their hopes too high. This system would not cure Linda, she would still need caretakers nearby at all time and she would still need instant help when having a seizure.

It would probably be difficult to let go of the existing system and replace it with a monitoring system like the LifeShirt system. It is often scary and difficult to let go of something familiar, and try out something completely new. Especially in this situation where it is a human being and potential life threatening situations. It might be possible to use both the new system and have the old up and running at the same time for a test period. This could ease the change.

Unfortunately, the improvements just mentioned could not be achieved at this point. The LifeShirt was too big and bulky for Linda and the system not yet reliable enough. However, changing the LifeShirt and performing more testing to the system is definitely manageable at this point. The fact that by doing so could improve the quality of life for a whole family, should be motivation enough to continue with this work.

9.5 OTHER USE CASES

Linda has a rare disease and for her case the system has many advantages, both for her relatives and medical personnel. Using resources and money to develop the system further requires more than one potential user. What kind of people could have the same advantages of the system as the people involved with Linda, if any?

9.5.1 POTENTIAL USERS

In discussion with Linda’s physician, several possible user groups were brought to the table. At his department, more than one patient has rare and unpredictable lung diseases. In cases where seizures could occur anytime, the system could be as useful as for Linda. The alarm function could lead to a safer environment for the patients outside the hospital, preferably at home.

For instance, premature born children are in high risk to develop breathing problems and could have advantages of being monitored. This would however require another monitoring shirt than LifeShirt, due to its size and heaviness. Connecting another shirt to the system should not be that difficult, as described earlier. If the only vital sign that needs monitoring is breathing, a light weight shirt might also be a cheaper solution.

These examples were only from the pediatric department. In general, this system could be used on people who are unable to take care of themselves or communicate with others. Mainly this is newborn, elderly and people with disabilities. Such a system could have them monitored all the time by relatives or others, and this could lead to a better life for the involved parties.

Use of the system needs cooperation between medical personnel and patients/relatives. The shirt is quite expensive and most people would need some training to take the system in use. This means that users and medical personnel need to be motivated and eager to use it. Although we only have feedback from one user case and one hospital, potential users of this system are people that have a reduced quality of life. If they get the opportunity to have an easier everyday life, why shouldn’t they be open minded and positive to such a system?

An important point is that you don’t need to change the existing system. The system is only based on requirements from the specific user case, but according to Linda’s physician the system has enough functionality and opportunities to use it on others (at least children) straight away. No adding of functionality, but fulfilling the requirements, is needed. This shows that extended use is realistic.

9.5.2 HOSPITAL USE

In addition to take the system in use inside a patient’s home, it could also be of use in a hospital. For instance is there no monitoring of breathing in hospitals today, and often patients need to have constant human monitoring. This requires resources that could have been used on other work. If patients used the system, medical personnel could do other work preferably nearby and be alerted as soon as something was abnormal. This requires training of medical personnel, but the resources needed for this might be worth it.

To give an example; when Linda’s parents needed assistance in taking care of her, they asked for her to be transferred to the hospital. They were told that she could stay in hospital, but the hospital had no resources to look after her. In other words, she could stay in hospital, but her parents still needed to take care of her constantly. Thus the point of hospitalizing her was all gone. This could all have been changed if a monitoring system like the LifeShirt system was taken in use.

Linda's physician did mean that in theory, they could take the system in use at the hospital immediately. In practice, there is a long way to go. Changing working processes is not done over night, but the fact that the simple system we made could be useful says a lot.

In addition to alarms, the saving of data is of huge importance at a hospital. This is discussed further in the next section.

9.6 USE FOR DIAGNOSING

All monitoring data are now saved in a database. The idea is that users could go back and look at monitoring data from interesting time slots. This applies to medical personnel as well. Today at hospitals, no monitoring data is saved. Patients at risk are monitored with machines, but this is only real time monitoring and medical personnel has no opportunity to go back and see what happened in the past. This is possible with our monitoring system.

Our monitoring system gives the opportunity to go back and see exactly what happened to the patient's body before, during and after a seizure. This could be interesting. To see what part of the body goes into an abnormal state first could give priceless information. Especially for patients with an unknown diagnose, where this could take the discussion around a diagnosis one step further.

Linda is a good example of this. Her condition is unknown and medical personnel do not know what causes her seizures. Her doctors are unable to agree upon what causes them because they all believe that she is healthy according to their special field. For instance her heart doctor is certain that her heart is normal and her lung doctor is certain that her lungs are normal. Saved monitoring data could tell what triggers a seizure and therefore help in diagnosing.

Monitoring data could also be shown to the patient or relatives to give them something tangible when discussing their disease. This could probably comfort patients and make medical personnel more trustworthy.

At this point there only exists a simple client with simple graph generation, but you could extend this with another client that gets more complex data in other ways. Also, if monitoring data should be used in diagnosing it is very important that the data are correct. If timestamps are slightly wrong this could lead to wrong conclusions. It is of importance that medical personnel using such systems know exactly how reliable the monitoring data are.

Linda's physician pointed out that in cases with unknown diagnoses, medical personnel know very little about vital signs values and in the start simple data as the ones in our monitoring system is more than enough. Having in mind that doctors are used to look at graphs from already existing monitoring screens in hospitals, our simple graphs would not require much training to use.

If simple monitoring data from our system reveals what vital sign causes abnormal behavior in a patient's body, medical personnel could go deeper in on that vital. In such situations it could be interesting to use more detailed monitoring data, like the software provided with the LifeShirt. However, using such a monitoring system at once would be too detailed. Use of advanced software will probably require more training and would be more expensive.

It is difficult to tell beforehand if it is possible to diagnose based on these kinds of monitoring data. However, compared to how it is today, it clearly gives new insight and new possibilities.

9.7 LIFESHIRT INTEGRATED IN THE NORWEGIAN HEALTH SYSTEM

Our focus was on creating a prototype concept. However, our system could be used in a larger setting in the future. Here we outline some thoughts on what could be the future.

A trend seen in the health sector today is an increasing focus on the patient. The patient should be in focus for all communication between different health instances and patient information should be centralized.

Today many health processes are performed on paper and not saved in computer systems. It is not many years back when your x-rays were printed and delivered by hand from the x-ray department to the examination room. More and more processes are transferred to computer systems, but an obviously weakness is that the computer systems are stand alone systems. Your private doctor has one system with information about you, the nearby hospital another, your therapist yet another, and so on. This is the reason why tests ordered by your doctor have to be sent by mail to a hospital for an analysis, and the results are sent back written on paper.

A centralized system would remove all this unnecessary work and also remove several sources to error. This system could offer services and different health instances could use only the parts they need. This can be seen as a service oriented architecture approach, which is a trend in other sectors today as well.

Our system fits straight into such a centralized system. Monitoring data is stored on patients and these data is reachable through services. Different health instances have access to the same data and could also see each others evaluations of the patient. The patient could have access to these data as well, and communication between medical personnel and patients would be easier.

This would lead to a different way of working for medical personnel. They could take advantage of each other and share knowledge. Several manually work processes could be removed and the focus could be completely on the patient. The patient would feel much more comfort knowing that their health history is available anytime anywhere.

A monitoring system could also free up hospital beds, if patients could monitor themselves or be monitored by relatives. Instead of being hospitalized, medical personnel could watch their patients' health through saved monitoring data. These data could also be a better fundament for diagnosing since the patient is living a normal life at home. In other words, this could be a win-win situation for both patients and medical personnel.

A centralized system for Norway is however a long way to go, there are many challenges that need to be addressed in order to make such a large system:

- There are massive amounts of already existing data that has to be entered into the system, from paper and different data systems.
- There will be enormous amounts of data and this has to be sorted and viewable in a clear and concise manner.
- Security concerns are very important when dealing with sensitive data.
- Work processes have to be changed, and new processes and procedures would be required.
- New organizational units, reorganizing existing units.
- All personnel would need training in the new system and the new procedures.
- Creation of the system would be time consuming, so the system has to be made in a upgradeable and extendable fashion so it won't get outdated quickly.
- Operations and maintenance.

In many ways Norway is more suited to implement a nation wide health system than other countries. There is a lot of focus on using technology in Norway's public sector, with an aim that as much as possible should be possible to do online. For instance Norway is one of the first countries in the world where the tax return is filed on the Internet. And due to the size of Norway it would be easier to implement a nationwide system than for other larger countries.

Our solution is a small step towards the future health sector. The patient is in focus and physicians may cooperate on the same centralized data.

10 CONCLUSION AND FURTHER WORK

"He who has health has hope,
and he who has hope has everything."
- Arabian Proverb

This chapter concludes our work and discusses how this work could be taken further. The conclusions are based on the research questions described in section 3.1. This chapter also outlines some future possibilities for a monitoring system like the one developed in this thesis.

10.1 CONCLUSION

The personal monitoring system developed in this thesis has mainly received positive feedback from the involved parties. Testing of the system in a real-life situation has given valuable feedback to use in further research within this field.

Our research question areas were twofold. One area was concerned with the possibilities to extend the monitoring system developed to suit other potential users, not implementing new systems to each new user. Based on the feedback from Linda's physician and our experiences developing the system, we conclude that this is indeed possible.

First, we have found that there are many potential users of our system. As described in section 9.5.1, Linda's physician pointed out some user groups from the pediatrics department, and we strongly believe that this goes for other departments as well. Alarm functionality would be useful to patients unable to take care of themselves and the data saving functionality could be useful to other patients with unknown diagnoses as well.

Second, Linda's physician believes that this system could be taken in use by many users only with the functionality present today. Nearly no modifications need to be performed. Several different users actually have the same requirements as our specific user case. The user has the possibility to change vital sign thresholds real time, and this is all that is required to personalize the system for now.

Third, we have developed the system with modifiability as an architectural driver, as described in section 5.1 and discussed in section 9.2.6, and the server part of the system should be relatively easy to change. In the nearby future there would probably be only minor changes needed to suit new users, and this should be a straight forward process. The system is easy to extend and a huge strength is the possibility to attach new shirts without modifying the whole server part. The same goes for clients; new clients can be added with no modifications to the existing system.

The other area within our research questions was concerned with Linda and her family's daily life. As described in detail in section 9.4, Linda and her family would benefit from a working personal monitoring system in several areas. Their lives would be easier compared to today, and they would have more freedom to move around and live like ordinary people. Unfortunately we weren't able to test the system in practice at this point, due to some critical weaknesses in the existing LifeShirt described in section 8.4. However, feedback from the involved parties leads us to conclude that the system could improve the health-related quality of life for Linda and her surroundings. According to her physician, it would also most likely help in diagnosing her which was described further in section 9.6. That is, if the system is made more reliable and clinical tested. Also, the shirt would have to be replaced by a more suitable one.

To see if this system was worth taking further we also compared our system with the existing system provided with the LifeShirt. The main problems reported from Linda's father were concerned with the LifeShirt itself. The battery capacity was too low, the shirt contained too many wires and the attached PDA was too heavy. Using the system provided with the LifeShirt would obviously have given the same problems.

However, looking at the software, the system provided with the LifeShirt and the system developed in this thesis has distinct differences. The system provided with the LifeShirt has limited alarm functionality and the data are stored in different files which have to be loaded into the application. In our system all monitoring data are stored in a database for easy access and comparing of data. With the already existing software, there is no possibility to use mobile devices and it is not possible to modify the program to suit different users better. Also, the doctor we have spoken to was clear in his statement that the provided system was too difficult and detailed to use on new patients. It would require a lot more training than with our system. Of these reasons it is safe to conclude that our system is an improvement over the standard solution provided with the LifeShirt.

The system developed has great potential both for Linda and other users. It does not need to be modified a lot, but some small improvements should be implemented. The most important is to test the system further and raise discussions with more potential users and medical personnel. We conclude that the work is worth taking further.

10.2 FURTHER WORK AND POSSIBILITIES

This thesis has taken personal health monitoring a step further, but a lot of work remains before this can be used widely.

Some work remains on the system to fulfill the requirements from the specific user case. To give the family fully potential of the system, these should be implemented; at least the one with medium or high importance.

To have Linda's family use the system further, a more suitable shirt needs to be studied. The problems they found with the LifeShirt would probably be the same for other potential users as well. The LifeShirt is not particularly suited for children, and children could be an important user group for this system. This new shirt might be the VivoResponder which was discussed in section 2.2.2, or a shirt from another company than VivoMetrics. The shirt has to be possible to connect to, and the data from the shirt has to be possible to transform to a format known to our system.

We recommend that monitoring shirts and other components out of the developers' hands are tested in the user's environment as early as possible. If the LifeShirt had been tested on Linda immediately after receiving it, it would have been discovered that the shirt was useless and another shirt could have been found before the implementation started. Knowing that the monitoring shirt suits the user also lets the user focus completely on the developed monitoring system during testing.

According to medical personnel the existing functionality should be enough to take the system in use on other patients as well. Now, the system is quite difficult to install and this should be changed. A difficult installation process could take away users' motivation and make them refuse to use the system. It is also of importance that the help functionality is implemented and is described correct and detailed.

This system is used in situations where there is a potential life danger. It is therefore crucial that the system is stable and bug free. So far the system has only been created to give a foundation for discussion with the specific user case and medical personnel. Debugging and making the system more robust should be performed as soon as possible; at least before testing the system on other users.

Feedback from only one doctor and one user case does not give a strong statistical ground. The system should be discussed with other medical personnel and tested on other user cases. This could reveal new requirements and useful feedback that our test persons haven't thought of. When having feedback from enough users, improvements to the system could be made.

When the system is robust and debugged sufficiently, the work is to convince hospitals to try it out on patients at home and maybe even in hospitals. This could be quite difficult, but a monitoring system could ease the everyday life for several patients and even simplify the medical personnel's work. If you could get people to understand the impact a monitoring system could have in life, it should be feasible to start experiments.

Presenting this thesis to Accenture Norway resulted in a lot of positive feedback. Their interest indicates that the company sees the potential in such a project, and hopefully this leads to further research with this monitoring system.

The future possibilities for an extension of the system are many. It can be used on a wide range of patients, from newborn to elderly. It can be used in patients' home, at hospitals and other places like for instance care centers for elderly. It can be used on patients with different diseases, known and unknown. It can be used to avoid life threatening situations and it can be used to diagnose patients. It can be used as a community where patients can share experiences and compare monitoring data. With a central sever it can be used in research and monitoring data from several patients with the same disease could be compared. In other words, a monitoring system with the same concepts as the one developed in this thesis could save lives, improve the quality of life, and give new knowledge to the medical world.

REFERENCES

Bagozzi, R. P., & Davis, F. D. (1992). Development and Test of a Theory of Technological Learning and Usage. *Human Relations, Vol. 45, No. 7* , 659-686.

Bass, L., Clements, P., & Kazman, R. (2006). *Software Architecture in Practice, Second Edition.* Addison-Wesley.

Guyatt, G. H., Feeny, D. H., & Patrick, D. L. (1993). Measuring Health-related Quality of Life. *Volume 118 Issue 8* , 622-629.

Haugros, H., & Overå, S. (2007). *Personal Health Monitoring.* Trondheim: IDI.

Inc, P. (2008). *WS-I Organization*. Hentet May 29, 2008 fra Welcome to the WS-I Organization's Web site: http://www.ws-i.org/

Jones, V., Robert, C., & Istepanian, S. (2006). Remote Monitoring for Healthcare and for Safety in Extreme Environments. *M-Health* , 561-573.

Korhonen, I., Pärkkä, J., & Van Gils, M. (2003). Health monitoring in the home of the future. *IEEE Engineering in Medicine and Biology Magazine vol 22.* , 66-73.

National Institute of Neurological Disorders and Stroke. (2007, December 11). *Chiari Malformation Information Page*. Hentet May 22, 2008 fra Chiari Malformation Information Page: http://www.ninds.nih.gov/disorders/chiari/chiari.htm

Paradiso, R. (2003). Wearable Health Care System for Vital Signs Monitoring. *IEEE Conf on Information Technology Applications in Biomedicine* , 283-286.

Tamura, T. (1998). Fully automated health monitoring system in the home. *Medical Engineering & Physics* , 573-579.

VivoMetrics. (2007). *VivoMetrics - About us*. Retrieved May 22, 2008, from http://www.vivometrics.com/corporate/about_us/index.php

VivoMetrics. (2004). *VivoMetrics LifeShirt System Technology*. Retrieved May 22, 2008, from http://www.vivometrics.com/docs/LifeShirt_System_Technology.pdf

Yeatman, E. (2006). Rotating and gyroscopic MEMS energy scavenging. *Wearable and Implantable Body Sensor Networks* , 4.

APPENDIX

The appendix contains information considered as extra material to give a deeper insight.

A. DATABASE DESIGN

This section presents the database model for the system and describes their entities. Figure 10-1 presents the database design.

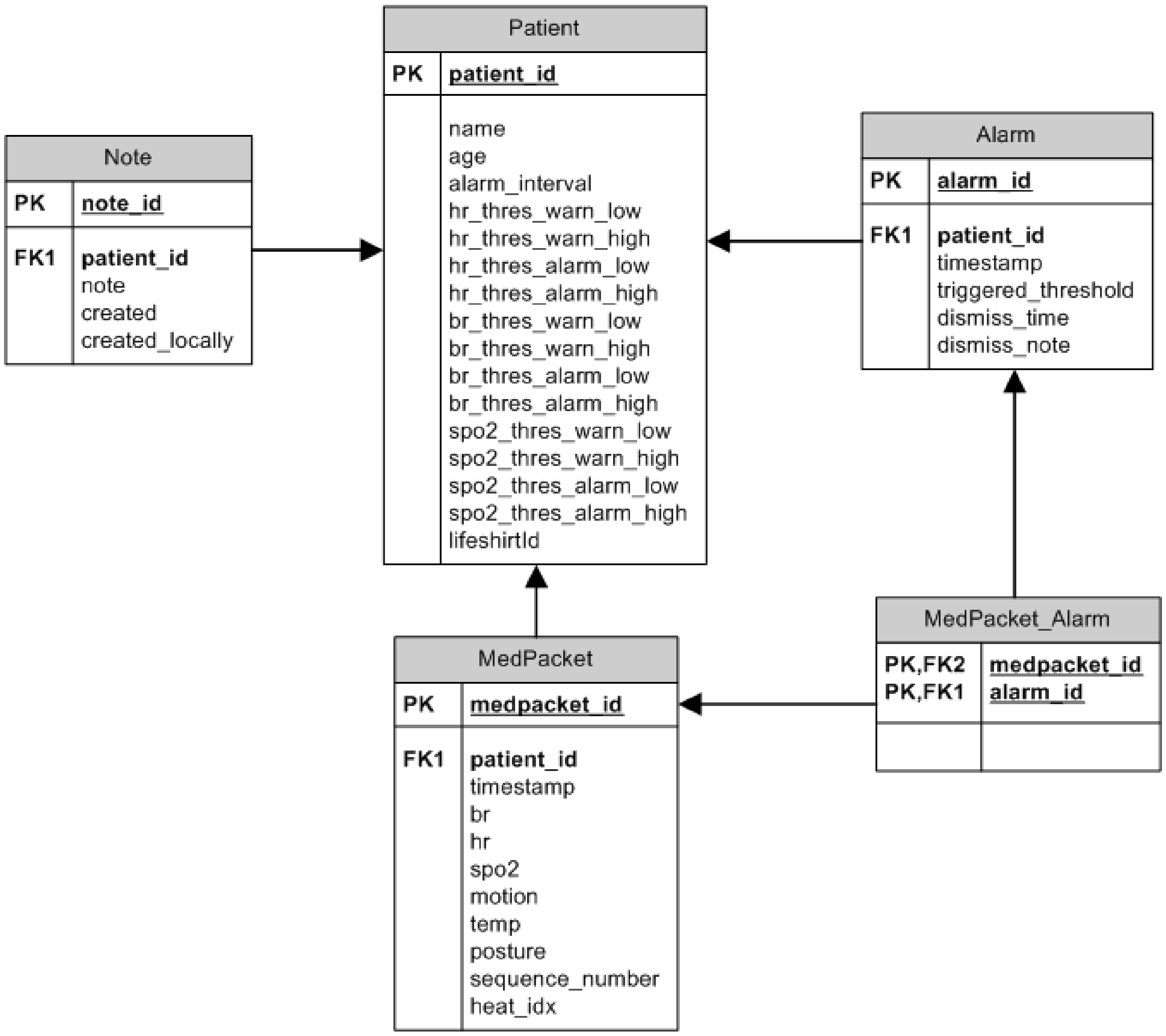


FIGURE 10-1 - DATABASE DESIGN

A.1 PATIENT

One patient is one user of the system. Users have to register themselves in the system and fill out thresholds for different sensors. Also, the lifeshirtid mentioned in 6.1.1.1 must be chosen. The alarm interval is an interval stating for how long monitoring data shall be stored before and after an alarm.

A.2 NOTE

A note is as the name implies, a note filled out by a user through the client. This is stored with a timestamp and can be seen together with monitoring data around the same date. It also contains a variable saying whether the note was created at a PC client or a mobile client.

A.3 MEDPACKET

Monitoring data from the LifeShirt arrives as text blocks which are processed and stored as medpackets. Each medpacket contains the different sensor values and the time when it was monitored.

A.4 ALARM

Whenever monitoring data is below or above the thresholds chosen, an alarm is created. The alarm contains the time of creation, and the threshold which triggered the alarm. Also, it contains a collection of medpackets from a specified interval before, the medpackets that are received as long as the alarm is active, and from a specified interval after the alarm is dismissed. The alarm is dismissed if the triggered threshold is a warning and the sensor value increases or decreases to alarm state, or when the user actively chooses to dismiss the alarm. You can dismiss the alarm both from a PC client and a mobile client.

B. INSTALLATION GUIDE

The following section provides detailed information of the setup and initial configuration of the system. This is a guide to set up the programming environment, a production environment is relatively easier to configure for the end-user. However, this was outside our scope.

The setup of the system can be done in multiple ways, with dynamic ips and the possibility for the clients to connect to the server from the Internet, the following details the most simple setup of the system.

B.1 SOFTWARE INSTALLATION

- *VivoMonitor 1.0* must be installed; this package provides a server component that may be connected to.
- *PostgreSQL* is the database that we use. It must be installed and configured with a database called 'lifeshirt' and a database user with access to this database must be created
- *Apache Tomcat* is the web application server we use, it is used by eclipse to provide web service functionality
- Create the folder *avatar* under C:\, this is where the MsgClient will output what it receives

Eclipse

1. Install the latest version of Eclipse J2EE.
2. Install the Hibernate package for Eclipse from JBoss
3. Do a full checkout from our repository, the main project should then be set up with the necessary jar archives on the build path
4. Modify *hibernate.cfg.xml* with the appropriate postgres username and password
5. In the 'Servers' view in Eclipse, add a Tomcat server
6. Right-click the LifeShirtWebService class and choose *Create Web Service*
    - Choose style "RPC/Encoded"
7. Now the server should be started with the WebServiceProject
8. In the class *Startup* modify the path so that it corresponds with the VivoMonitor installation directory
9. Start *Startup*, the server starts the MsgClient which connects to VivoMonitor, and MedPackets should now be received by the server

PC-client

The PC-client is compiled to connect to localhost. Thus if it is run on the same computer as the server no recompile is required. Otherwise the web reference in the PC-client must be updated, and the client compiled.

Mobile Client

Since the mobile client connects over a network carrier to the server, it has to be recompiled when the server is fully configured. Update the web reference with the ip to the server, and then do a recompile. The MobileClient can then be transferred to a mobile.

B.2 PHYSICAL INSTALLATION

Server

The server should be installed in the local network with a static ip. A static ip makes it easier to configure the LifeShirt and the mobile client. Optimally the server should be configured with a name in a name service, but this service is usually unavailable on home networks.

The LifeShirt
There are three parameters that have to be set in the LifeShirt configuration files, these files are found on the memory card that is inserted into the PDA. When editing make sure to use a program that handles UNIX style newlines, meaning do not use Windows Notepad.

- In *net.con* enter the ip address of the server
- In *wireless.opt* enter the network name (ESSID)
- In *wireless.opt* enter the ip the LifeShirt should acquire from the network (IPADDR)

MobileClient
Connect the MobileClient to the same network as the server.