INF-3996

MASTER'S THESIS IN

TELEMEDICINE & eHEALTH

A Context-Sensitive Framework for Mobile Terminals for Assisting Type 2 Diabetes Patients

Taridzo Fred Chomutare

June, 2008

FACULTY OF SCIENCE
Department of Computer Science
University of Tromsø

INF-3996

Master's Thesis in

Telemedicine & eHealth

A Context-Sensitive Framework for Mobile Terminals for Assisting Type 2 Diabetes Patients

Taridzo Fred Chomutare

June, 2008

Dedicated

With Love To

My Parents

Preface

The thesis is intended for researchers interested in patient-operated disease management tools on mobile phones. The research is useful for developers and architects alike. For the architects, the thesis presents a framework for constructing context-sensitive applications while at the same time the developers will benefit from the exemplified solutions to practical problems encountered when modelling context on mobile phones.

The underlying motivation was that patients abandon disease management tools within short periods. The actual cause for this behaviour is uncertain and some researchers propose that usability issues are a major contributing factor. It could easily be boredom as well, but the use of contextual information to increase usability can be examined to determine whether enhancing usability will help the situation. For this research, usability is defined loosely and includes user/patient satisfaction with, effectiveness and efficiency of [1], an application based on the context-sensitive framework.

This thesis is based on on-going research at Norwegian Centre for Telemedicine (NST), which is a branch of the University Hospital of Northern Norway (UNN), in the “Lifestyle” group. The group develops tools and components for self-managing Type 2 diabetes and has already developed a prototype to demonstrate the usefulness of self-help tools [2]. A wireless blood glucose sensor was constructed to automatically transfer blood glucose measurements to other electronic media such as mobile phones and computers [3], and thus enabling the possibility of integrating with a remote Electronic Health Record (EHR).

Physical exercise and nutrition are important determinants of blood glucose levels for Type 2 diabetes patients. Therefore, the group constructed a “step-counter” which uses the number of steps as a measure of physical exercise [4]. The step-counter is wireless and automatically transmits the number of steps taken per given intervals using a Bluetooth adapter. For nutrition management, the group has designed an electronic diary as part of an integrated package for the prototype [5].

The prototype and its components are mobile-based, and my thesis builds on them; to make them more usable for patients. My research sought to add enhancements to the prototype and platform in order to increase usability. The prototype will be deployed in the near future and it is hoped patients will retain use of the resulting application over long periods or until a healthy lifestyle is adopted and maintained.

I would like to thank the people who made the research a resounding success. Most of the people are members of the research group at NST and the University of Tromsø. I appreciate NST’s and the university’s assistance with the project equipment; the laptop, office space, and mobile phones. The Norwegian State Educational Loan Fund, Lånekassen, financed my studies and I am eternally grateful for their support.

I would like to thank my supervisor, Prof. Gunnar Hartvigsen, without whose expertise, guidance and support the project would never have been a success as it is this day. I am grateful for his selfless devotion and time he used for grooming me, despite his busy schedule. I learned a great deal from his wealth of experience, passion for research and dedication to his responsibilities.

I am most indebted to my co-supervisor, Eirik Årsand, on whose work my project is based. He has been my source of inspiration, continually steering me in the right direction. His stimulating suggestions, timely feedback and mentorship are second to none. I was motivated and given tenacity by his display of concern over my personal development and career growth.

My profound gratitude goes out to Ragnhild Varmedal for her outstanding role in the project and giving me my initial steps towards programming for mobile terminals. I greatly appreciate her tireless dedication to my project needs. I thank Naoe Tatara for her valuable feedback and proof-reading during the design, implementation and experimentation. I also thank Niklas Andersson, Harald Øverli Eriksen and Luis Fernandez Luque who consistently offered tips and information related to my research.

Lastly, but by no means least, I thank my parents and siblings for their moral support through encouraging phone calls, SMS and emails.

Most importantly I thank the eternal, most blessed God, creator of the universe, who fathoms all contexts. What would I do without Him!

Abstract

Purpose The aim of this research is to develop a framework for using context information to create an intelligent environment within which self-help mobile applications reside. The concrete application is to use the context information to enhance usability of a diabetes self-help tool [2] that was designed for improved disease management.

Motivation Mobile applications are emerging as a preferred method for diabetes self-management [6]. This increases interruptions that the patient will get from the phone and can easily become annoying. Using context information holds a potential to increase the usability of such tools. For instance, convenient delivery of message alerts reduces obtrusiveness, making it easier for the tool to be accepted as part of the patient's life. Since specialised sensors are still not widely available, an alternative is to creatively exploit potentially useful context information sources within a mobile phone.

Methods An engineering approach was used for systems design, resulting in a prototype that uses a step-counter, calendar, microphone, camera, battery and onboard clock as sources of context information. These sources of potentially useful contextual information have been widely available on ordinary mobile phones for a while, except for the step-counter which is only recently becoming available.

An inference engine was designed using an event-driven architecture while context was modelled using a generalizable abstraction comprising 2-state finite state automata. Emphasis has been on efficient algorithms suitable for mobile phones and rules based on sound logical deductions. Rule priorities were based on the results of a survey conducted on 11 diabetes patients. In addition controlled experiments were conducted to test the accuracy of context detection and the use of resources on the mobile phone.

Results The framework provided an efficient way of detecting contexts within an acceptable magnitude of accuracy. The effect on mobile resources was almost insignificant; with effect on battery less than 15% and occupying less than 1% of main memory. The phone could detect some simple situations and adjust the phone accordingly. However, when considering ambient light and noise, the situation became more complex. For example, a finger covering the camera would result in the phone assuming it is dark. Dealing with such uncertainties may require additional context information or specialised sensors.

Conclusion The developed context-sensitive framework was able to use context information to create an intelligent environment that assists Type 2 diabetes patients by enhancing usability of self-help tools on mobile phones. Modelling contextual information using the proposed generic abstraction and an event-driven architecture was suitable for mobile phones because it lowered time and space complexity and solves the problem of modelling context from heterogeneous data sources. Research projects based on multiple specialised sensors, complex data structures and intensive computation are unlikely to be successful on current ordinary mobile phones.

Table of Contents

List of Figures

List of Tables

List of Code Snippets

Notations/Abbreviations

API	–	Application Programming Interface
BAN	–	Body Area Network
CS	–	Context-Sensitive (Context Aware)
ECA	–	Event Condition Action
EHR	–	Electronic Patient Record/Electronic Health Record
FSM	–	Finite State Machine/Automaton
GB	-	Gigabytes
IDE	–	Integrated Development Environment
KB	–	Kilobytes
MB	–	Megabytes
NST	–	Norwegian Centre for Telemedicine
NST	–	Norwegian Centre for Telemedicine
OEM	–	Original Equipment Manufacturer
PDA	–	Personal Digital Assistant
PDMS	–	Personal Diabetes Management System
POOM	–	Pocket Outlook Object Model
RMS	–	Root Mean Square
SMBG	–	Self-Monitoring of Blood Glucose
SMS	–	Short Messaging System
SRE	–	Simple Rule Engine
TEA	–	Technology for Enabling Awareness
UML	–	Unified Modelling Language
XML	–	Extensible Markup Language

CHAPTER 1

Introduction

1.1 Background and Motivation

Managing Type 2 diabetes involves patient education and personal effort from the patient to improve or adopt a healthy life style. Research has shown how constant self-monitoring of blood glucose (SMBG) and education can make a huge difference [7] to the patient's health. However, the benefits of SMBG are not so apparent in patients who do not use insulin if they take no action on the feedback from the Personal Diabetes Management System (PDMS) because of ignorance [8], negligence or otherwise. Benefits can be accrued if the patient is educated enough to know what action to take in different feedback scenarios [9].

Common PDMS tools include an electronic diary [10] for nutrition and physical exercise management, and educational tools to empower the patient with knowledge. Use of such tools implies more user interaction with the phone. This research seeks to make such PDMS tools more usable by considering contextual information when interacting with the patient. Using context-sensitivity to improve usability of the tools is a promising technique that holds a potential to increase the usefulness both for people with Type 2 diabetes and other chronic diseases. The important challenges include developing a generalizable framework for modelling, handling and using the context information.

Informatics tools used to help manage Type 2 diabetes have mainly been web-based [11-14]. However, things are changing with the advent of mobile technology. Several attempts have been made to develop disease management applications for mobile phones. Wide spread adoption of mobile-based applications has been limited and patients are known to abandon the applications in the long run [15]. Other mobile-based applications have been a total failure [16]. The mobile phone has a small screen and this places certain limitations on interaction design. The small keypads also present opportunities for erroneous input and ultimately, frustration. Again there will be some considerable increases in messages that the patient will receive. This, coupled with communication with healthcare providers, family and friends, will increase the activity on the mobile phone. If left uncontrolled, the calls and messages will become intrusive and the patient is eventually annoyed.

In addition, mobile phones have limited processing speed and battery capacity. In some instances, a separate device dedicated to monitoring health status has been prescribed. This means the patient has to have at least two mobile phones, one for communication services and another dedicated for healthcare purposes. Such a scenario is unsustainable, resulting in abandonment of such systems within a short period.

In a bid to develop applications that can be used for chronic illnesses, it seems worthwhile researching on how we can increase usability of chronic disease management applications. This is because chronic illnesses persist throughout the patient's life and use of such systems should be aimed at longer term commitments rather than short term fads.

As such, the applications must easily fit into the patient's life and be as unobtrusive as possible and become a natural part of the patient's life. I seek to make such tools more usable and integrate the applications with normal phone communication services. I try to make the mobile phone more "aware" of, or sensitive to, its environment and thus adapt the self-help tool services on the mobile terminal to the patients' needs. The phone should be able to anticipate the patients' needs as dictated by the environment and be "smart" enough to tell if the patient can take a call or message or if the patient should not be interrupted at all.

1.2 Scope and Research Problems

The research problem can be summarized in the following statement:

"*How can a context-sensitive framework on mobile terminals be constructed and structured to assist Type 2 diabetes patients self-manage the disease?"*

In the following sections I dissect the above problem statement in order to clearly define the boundaries and scope of this thesis. By subdividing the problem statement I derive five sub-problems (A-F) stemming from the main components of the problem statement as given above.

A. Context Information Sources

Normally, context information can be obtained from various specialised sensors. Unfortunately, these are still not widely available on ordinary mobile phones and may not be available in the foreseeable future. Most specialised activity sensors and sensors for ambient intelligence are still under research. An alternative approach encourages creativity when identifying potentially useful sources of context information and this involves analysing otherwise unlikely sources.

The emphasis of this research is on innovatively identifying as many context information sources as possible on the mobile phone. The identified sources should be relevant to interaction between the patient and the PDMS on the mobile phone. I can therefore articulate the first sub-problem as follows:

Question 1: What context information sources are easily available on mobile phones and are also relevant to managing Type 2 diabetes?

B. Context Modelling and Data Representation

A great challenge is how we should extract and model context information from the identified heterogeneous context information sources. The designer must be innovative and be able to use the identified context variables and develop realistic abstractions. This process is complicated by the fact that there is no universally accepted format or standard for representing contextual information. Many researchers have done work on modelling context information and they each have their different proposals based on their experiences and situations.

The reason why it has been difficult to develop context-sensitive applications widely could be the absence of a widely accepted architecture or framework for constructing such applications. Despite all the research effort, context-sensitive applications remain unused on a wide scale. It will be some time before generic platforms and standards are established. The purposes and use of context information are diverse and case-specific and depends on the researcher’s perspective, objectives and goals. Therefore, it seems worthwhile developing models geared towards generalizable situations and use cases. This leads me to the next major sub-problem:

Question 2: How can context information be extracted and modelled using generalizable abstractions?

C. Framework Architecture

Mobility inherently implies shifting between different contexts rapidly. When relevant context information has been identified and after correct abstractions are made, it is important to use the information to adapt the mobile terminal services to the situation. Elaborate and computation intensive approaches will generally not be feasible in the foreseeable future unless processing speed and battery capacity are improved drastically. A good approach uses as little battery power as possible and also implements efficient algorithms. While not compromising basic functionality, it is important to keep it simple in order to lower computation complexity in terms of both time and space.

Adaptation is the most important component because it describes how the context information is used to adjust behaviour of the self-help tool and other phone applications. Adapting applications also involves checking against patient preferences in order to more meaningfully assist and predict patient needs.

The techniques for determining intervals for checking context variable states are important. For example, the framework can poll for state changes and update the phone. Alternatively, the framework can wait until certain named events occur before adapting the phone. These issues constitute the overview of how the context-sensitive framework will operate and can be summarized in sub-problem three below:

Question 3: How can the modelled context be used to adapt the self-help tool and the terminal’s communication services to changing context?

D. Limited Mobile Resources

Mobile terminals have limited resources and therefore it is important to rationalise and optimise use of such resources. The following are some important resources that are critical and must be used efficiently:

- Battery - Memory - CPU

The battery is an important resource that must be conserved. Usability diminishes if the patient has to recharge the battery more often than is necessary. Main memory is important to consider because the diabetes management tools do not run in isolation. They co-exist with other applications that also compete for the same resources. It is important that the applications share the available memory without some starving other applications. The same applies for CPU time slices that have to be shared among the different applications. It is important therefore to avoid unnecessary computation, which will also result in battery consumption, hence the question:
Question 4: What are the design considerations for efficiency of a context-sensitive framework on mobile terminals?

E. Assisting Type 2 Diabetes Patients

It is important to note that this research does not deal with creating a Type 2 diabetes self-help tool, but rather describes a framework for making the self-help tools more usable, with the example of the NST prototype described in [2]. Type 2 diabetes management in most cases does not involve the use of insulin. The benefits of SMBG depend on the patient's ability to know what to do, for example, when glucose measurements show a certain trend. Most of the self-help tools are geared towards patient education and this implies more time to interact with the mobile phone.

It is therefore imperative to incorporate features that are designed to make the tools enjoyable to use in different situations and avoid any form discomfort. The problem is what enhancement features can be added to the tools to make them easily adaptable to changing contexts and this begs the fifth question:
Question 5: How can the framework be used to enhance Type 2 diabetes self-help tools?

1.3 Assumptions and Limitations

The major assumption is that the framework is based on an ordinary mobile that has basic functionality, including an electronic calendar that can be synchronised and a simple camera. I assume the mobile is operated by one person, so that privileged information can be accessed without re-authentication. Another important assumption is that all the necessary components are open to the developer and not hidden by the original equipment manufacturers (OEM), for example low level hardware APIs such as camera operations and backlight implementation details. Another important assumption is that an adjusted step counter is available that can detect if the patient is walking, stationary or jogging. In this thesis I simulate the process using hypothetical data because a step counter is still uncommon.

1.4 Methods

I use an engineering approach as described by Denning, et. al. in [17], where I construct a prototype to demonstrate the solution to the problems. After the prototype is implemented I deploy the context-sensitive framework and conduct controlled experiments to determine the prototype's accuracy at detecting contexts and its effect on limited mobile resources.

Initially a survey is conducted to determine the target (diabetes patients) group's preferences and views on the different context situations. The mean age of the group was over 55 years and this has implications on values and priorities placed on different contexts. For example, it would make sense to prioritize computations for adjusting backlight intensity because the patient's eyesight has probably deteriorated and is sensitive to ambient lighting.

The short research time limit and legal restrictions did not allow for field tests on actual patients. This lack of feedback from actual patients may undermine the value of the results. However, some testing was done by colleagues and their feedback gave me important pointers. In addition, controlled experiments are thorough and reflect on the framework's behaviour on contexts that rarely occur.

1.5 Significance and Contribution

This research takes a generic approach and introduces simplified algorithms and techniques for modeling context information. The research, therefore, contributes significantly to making context-sensitive applications easier to construct and integrate with disease management tools. The resulting prototype demonstrates how applications and services can make use of context information to adapt to the patients' situation and preferences. In the next section I describe a hypothetical situation that demonstrates how a context-sensitive application could be beneficial.

1.5.1 Scenario

Uncle Bob has diabetes and is using a personal diabetes managements system (PDMS) on his mobile phone. He exercises at least once a day by taking a brisk walk. The PDMS sends information, but Uncle Bob is unable to hear the message alert until the evening when he is measuring his blood glucose levels. Just before he goes to bed, he receives a tip and some feedback on his performance that week. It takes him some effort to read because the lights in the room are already turned off and the mobile phone's back light is set to full intensity. He is excited about the positive reinforcement he got from the application but forgets to recharge his phone battery that night. In the morning, he gets a call while he is in a meeting with his boss. The alert volume was high because he forgot to tone it down when he got to work. His mobile switches off later in the day. Consequently, he forgets the doctor's appointment that evening.

This is an over dramatised situation, but it only serves to illustrate a point. A context-sensitive application would have known the following:

- Uncle Bob is walking and a detailed and graphic tip is not suitable.
- It is a noisy environment and he might not have heard the alert
- The room is dark and dimmed the mobile phone backlight appropriately
- The battery is too low to last the whole day tomorrow
- Bob is in a high priority meeting and should not be disturbed at all, or
- Bob is in a quiet environment and adjust the alert volume appropriately
- The battery will not last, and could have delivered the important appointment reminders *before* the battery went flat.

1.5.2 Summary of Goals

The prototype should be able to detect useful contexts and adapt the mobile to changing contexts. Accurately detecting context means the diabetes self-help tool can also be correctly adapted to changing contexts. Therefore, the framework should enhance usability while having an insignificant impact on resource consumption. Below is a list of my main goals which are based on the sub-problems discussed in earlier sections.

I. The thesis should stimulate creativity and innovativeness when identifying and extracting context information from otherwise unlikely sources.

II. The thesis should demonstrate how any proposed context modelling methods could offer generalizable abstractions of context information, and how efficiency is guaranteed when using the methods in resource-starved environments such as mobile terminals.

III. The thesis should demonstrate a concrete model or architecture for processing context information. It should clearly show how the model provides an easy way of handling the modelled context information to adapt the phone and the self-help tool efficiently.

IV. The thesis should prescribe a mechanism for completely separating the application code from the actual rules in order to create an opportunity for easily customizing and maintaining context-sensitive applications based on the framework.

V. The thesis should describe details for enhancing self-help tools that would make the tools more adaptable to changing contexts.

VI. The thesis should ultimately end up with reusable and extensible components to benefit future developers.

1.6 Organisation

The thesis is organised into the following chapters:

Chapter 1: Introduction

Chapter 2: Theoretical Framework
This chapter gives a review of related literature and state of the art, and attempts to summarise and classify aspects of the different approaches used by different researchers.

Chapter 3: Methods
The methods chapter describes the research methodology, the research paradigm used and the methods used for implementing and evaluating the prototype. It also acknowledges potential flaws in some methods used and offers a general critique.

Chapter 4: Requirements Specification
This chapter analyses the requirements and specifies the actual functional and qualitative requirements. It also describes the sources of the requirements and explains why some requirements had been rejected.

Chapter 5: Design
The design chapter details the design process, how context information would be extracted and modelled. It also describes how the experiments were conducted.

Chapter 6: Implementation
The prototype's implementation is described in this chapter, including the details of extracting context information from the different sources. The chapter also describes technical challenges encountered and includes elegant code for different computations.

Chapter 7: Discussion
The first half of this chapter presents the general findings and specific results of the experiments while the second half provides analyses and interpretations of the results.

Chapter 8: Conclusion and Future Work
This is the last chapter and concludes the research and offers suggestions for future work.

CHAPTER 2

Theoretical Framework

In this chapter I present the state of the art in diabetes management using eHealth tools. In the first section I discuss context and context sensitivity in general, and then later discuss generally the applications for diabetes management. The two form the main branches of this research and I also review the most recent literature leading to, but not including, my own ideas.

Search Criteria

In order to find out the state of the art, I searched several journals for related review papers. The three main databases of articles I used were PubMed, ScienceDirect, ACM portal and IEEE Xplore. There are two main combinations of search strings that I used:
I. Context-Aware AND Mobile
II. Mobile AND Diabetes

I used the AND logical operator as the conjuncture because a search for "diabetes management" alone would have many hits that are not informatics based but are only clinical or medical. Context-sensitivity is strongly associated with mobility and activity, and it is almost redundant to include "mobile" in the first search string. Generally, "context-aware" would yield more relevant articles than "context-sensitive". This led me to research further on the terms and their impact on the ultimate meaning.

I have found that papers may refer to context-sensitivity as [18]:

- Context-aware
- Ambient-aware
- Adaptive
- Responsive
- Reactive
- Situated
- Environment-directed

I may use these terms interchangeably. All the terms are strongly related to ubiquitous computing, pervasive computing and ambient intelligence. A prominent feature of these themes is the presence of computers everywhere but absorbed into the users' environments and transparent to the user, hence also overtones of human-computer interaction.

2.1 Context and Context Sensitivity

Context can be a very general term and is defined by HarperCollins dictionary[19] as:

"The words before and after a word or passage in a piece of writing that contribute to its meaning"
Or

"The circumstances relevant to an event or fact"

The first definition is often used in linguistics while the second definition is general and easily usable in computing. The second definition is so general that it makes it difficult to fully comprehend the meaning. "Circumstances" or variables that characterize a situation or setting are too many to comprehend or interpret for human beings, often resulting in misperceptions and misunderstandings. For computers, the situation is worse because of their limited ability to decipher some cues and subtle signals, moods or emotions. In the next section I give the definition as described in context-sensitive applications literature.

Definition: Context

Early research efforts concentrated only on location as the prime context variable [20], although later Schmidt [21] reveals in his paper that *"there is more to context than location"*. Another early attempt to define context was done in 1994 by Schilit [22] where he referred to examples rather than attempting to give a formal definition. Some of the examples he gives are *"…lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation…"*

Brown [23] narrows the definition only to include *"elements of the environment that the user's computer knows about"*. This definition limits the variables to only those a computer can process and thus precludes the many complex context variables that a computer cannot process. It is impractical to discriminate the variables purely on the basis that they cannot be processed by a computer and yet such variables may affect user interaction. Researchers must simply acknowledge that using context may be futile in some situations, especially if the complex variables have the greater effect on user interaction and yet they cannot be reasonably modelled by a computer.

A definition proposed in 1998 by Dey [24] included the user's *"emotional state and focus of attention"*. The definition thus explicitly ignores other factors that may affect the interaction, if the factors are currently out of the user's focus or are in the user's sub-conscience. There are many variables that exist that cannot be comprehended by the patient's cognitive power concurrently, and yet they affect how the patient interacts with an application. As if to correct this seriously erroneous definition, Dey and Abowd come up with a more concise definition in 1999 [25]:

"*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*."

This definition points out something very important, that is, the relevancy of the information. There are more variables that can characterize a situation than should be considered in any application. Therefore, the definition limits context to only relevant aspects and this is well suited to the software development field and also aligns with the formal dictionary definition given earlier.

Again, while this definition will be sufficient for my research, in practice there are still many relevant variables that cannot be incorporated in application design. For example, it may be difficult to always correctly model the emotional state of the patient or cues that indicate the patient is in a bad mood. It is also difficult to construct sensors for extracting and modelling such complex information as anger or excitement.

Definition: Context-Sensitive Application

The definition of a context-sensitive application is derived from the definition of context as discussed above. Generally, context-sensitivity can be defined as *"… the ability of a program or device to sense various states of its environment and itself" [26].* The statement does not state how the context is sensed or how it is used and this looks deliberate because it is left open and generic. This allows many types of designs to embrace such a definition. A more elaborate definition, however, is given as [18]:

"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

Therefore, context-sensitive applications are applications that are able to extract and use or provide context information or services. The context information or services are used to predict needs, adapt applications or components and generally better the interaction with the users. Humans can decipher most contexts easily by paying attention to things like non-verbal communication and cues, but the situation is different for a computer. The limiting factors are the sensor systems' ability to acquire and decipher complex context information. In the next section I describe the examples of eWatch [27] and ContextPhone[28], which are context-sensitive research projects and mention the kind of context information they use and how they use it.

2.1.1 Examples of Context-Sensitive Systems

Example 1: eWatch [27]

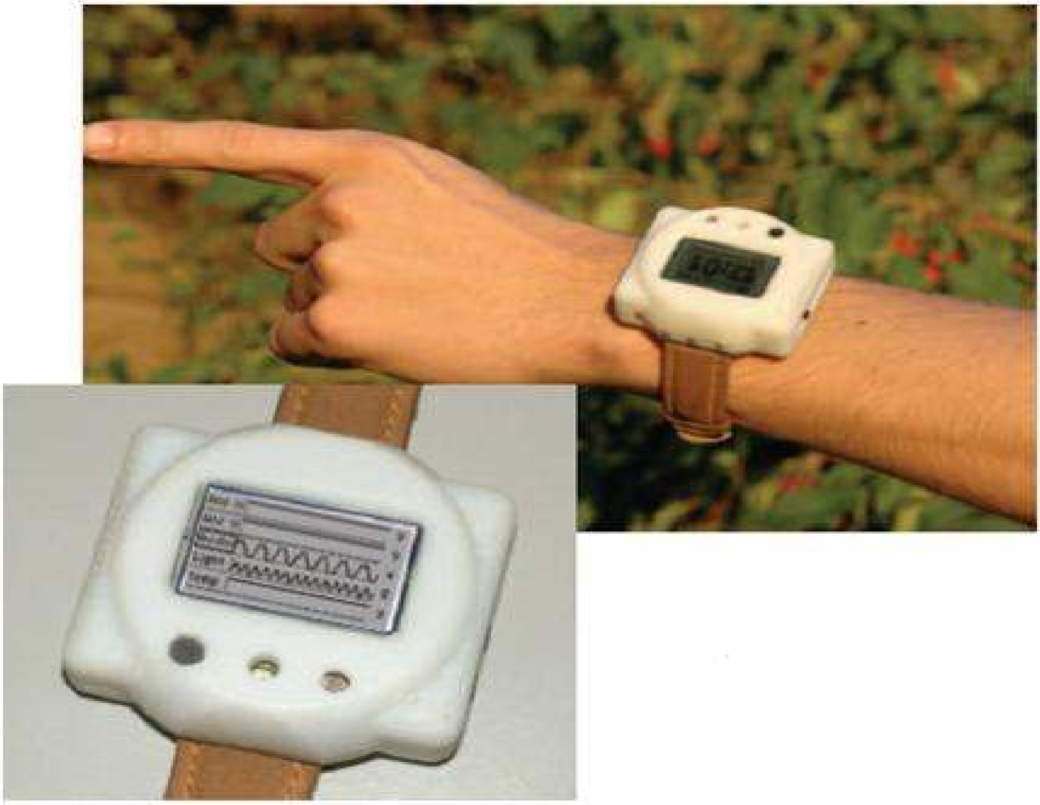


Figure 1: eWatch picture (Source: Fig.1 in [27])

The eWatch project presents a wearable watch that communicates with a PDA and email server. It takes a practical approach by using a non-intrusive and comfortable device the shape of a normal wrist watch as shown in the figure above. The project uses the following context information:

- Microphone (to get ambient noise)
- Accelerometer (physical activity)
- Light sensor (to get ambient light)
- Temperature sensor (body)
- Calendar (to get the busy status)
- email services (to get high priority emails)

The sensors are custom built for the project and uses unique protocols and data formats. The project also uses context sensitive output channels:
- LEDs (red and green)
- Vibration
- Ring tone

The eWatch looks like a simple wrist watch when infact it is an integrated sensor controller board. The sensors I described above, except for calendar and email services, are all embedded on the controller board and communicate with the PDA using Bluetooth. The PDA communicates with the email server and retrieves priority information from the email messages. The system's overview is illustrated in the diagram below:

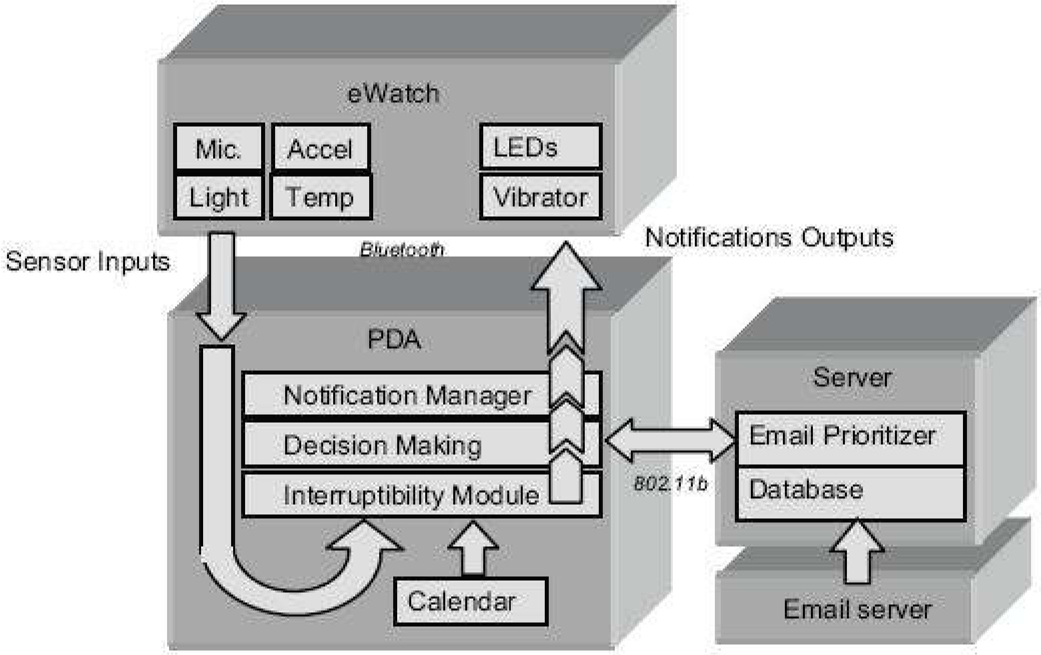


Figure 2: eWatch architecture (Source: fig.5 in [27])

The eWatch system calculates interruptibility of the user before using the context sensitive output channel. The system also determines the type of activity the user is engaged in and offers a classification of activities and places the user might be at:

- Walking
- Sleeping
- Lecture hall
- Cooking
- Reading
- Library
- Eating
- Office

The system is meant to be used by any one and the emphases are placed on context sensitive output channels. The user has to have the eWatch and the PDA as part of the system. The system uses IEEE 802.11b therefore the PDA has always got to be in range of the email server in order for the whole system to function properly. Although the system does not fully integrate with PDA phone (SMS/MMS and calls) functionality, for many practical purposes, this system is genius.

Example 2: ContextPhone [28]

The ContextPhone works in a similar way to the eWatch, but a fundamental difference is that all the sensors are embedded in the phone. This means the user does not need an extra technology gadget. Some of the context information that it uses is given below:

- Sound recorder
- Battery status
- Nearby device enquiry
- GPS
- Communication behaviour (calls and calls attempts)

Again, ContextPhone is integrated with some normal mobile phone functionalities such as SMS/MMS. The system offers context as a resource or service to other applications. This unique approach makes it possible for other application to use the context services that it shares. This was possible because the researchers do not use any custom built sensors to collect context information. Most of the context information can be extracted using a PDA's standard resources. However, the paper is not clear about how it uses all the mentioned sensors or context information sources. It also does not prescribe any generic models of the different context sources.

These and related systems have mainly been for experimental purposes. We still have not seen any serious or wide adoption of the concepts covered by such innovative ideas. We are likely to get mobile phones embedded with integrated sensors in the near future. For example, we already have a number of Nokia phones[1] with an accelerometer or activity monitor [29] and GPS.

[1] http://research.nokia.com/projects/activity_monitor

2.2 Taxonomy of the Concepts

In this section I discuss the different classifications of issues related to context-sensitive applications. Beginning with the context information itself and moving to classification of context-sensitive applications.

2.2.1 Classification of Context Information

There are different types and classes of context information and there are also many ways of classifying the information. The lack of universally agreed on classifications or standard indicate that this is a rich research field. The chaos at the moment, however, is becoming ridiculous because researchers end up with "*classifying context classifications*" [30] because of the too many classification possibilities. The different classifications often overlap, but below is a list as suggested in [31] and the corresponding examples:

- *Temporal information*; time of day, week, month or year
- *Environmental information*; light and noise level
- *Availability of resources*; battery, net work or bandwidth
- *Physiological measurements;* glucose data, ECG
- *Schedules;* as taken from the calendar program
- *Spatial information*; location
- *Activity*;talking,walking, running
- *Identity*
- *Social situation*; people nearby

The above classification is likely to overlap with any other classifications that exist. The list only serves as an indicator of what a typical classification can look like. Table 1 shows some example projects and the type of context information they use, utilising the preceding classification:

Table 1: Classification of context information used in different projects

Project	**Tempo**	**Env**	**Resource**	**Physio**	**Activity**	**ID**	**Spatial**	**Social**	**Score**
Stick-e [23] [32]	X	-	X	-	-	-	X	X	4/8
SmartReminder [33]	X	X	-	-	X	-	X	X	5/8
MyExperience [34]	-	-	X	-	-	-	X	X	3/8
eWatch[27]	X	X	-	-	X	-	-	-	3/8
SenSay [35]	X	X	-	-	X	-	-	-	3/8
TEA*	X	X	X	X	X	X	X	X	8/8
SenseWear [36]	-	-	-	X	X	-	-	-	2/8
IST Vivago [37]	-	-	-	X	X	-	-	-	2/8
ContextPhone [28]	X	X	X	-	-	-	X	X	5/8

*Several projects (Technology of Enabling Awareness)

The table above summarizes some projects and the type of context information they consider. The type of information generally depends on the aims of the project and the available sensors. In most of the projects the sensors are custom built. Integrated sensor chips or devices are generally not yet available for wide use; therefore, most of the projects build their own hardware and onboard circuits.

2.2.2 Classification of Context-Sensitive Applications

It is hard to find universally accepted classifications for context-sensitive applications. Each research projects seems to present its own classification rather than building on past work, depending their focus. This is making it difficult to come up with standard categories. Context-sensitive infrastructures are currently strongly tied to specific needs and situations. This state of affairs is not surprising since it is hardly a decade when intensive research in the area began.

Main ideas in related literature centre on wearable devices that are embedded with several sensors and transmit the patient's context information to a smart phone [38] [35]. In addition, information about the status of the device itself is also used and regarded as context information. To overcome resource limitations on mobile terminals, some researchers prescribe a central server where data is processed before it is served back to the mobile terminals through the internet, Bluetooth, WIFI, cables or otherwise.

In this section I discuss potential classifications of systems that take into account context information when interacting with users. In general, the design of such systems were broadly classified by Schilit as driven by either manual or automatic settings [22], although in practice both types overlap. I use this wide and general classification to categorize the different approaches taken by different research projects.

Manual Settings

The approach in this category is to provide the user with choices that are relevant to the context. The recommended choices are made more conspicuous and are based on context information. The user then chooses his or her preferences manually, which may not necessarily be the recommended choice by the system. For instance, if the patient needs to print an educational tip from the mobile phone, the system will ask the patient to choose from the list of nearby printers and making default the nearest printer.

One major disadvantage of this approach is that the user always has to be prompted before any action can take place. This can be quite annoying especially if the user changes environments frequently and every environment posing questions. Such applications are referred to as *"proximate selection"* and *"contextual command*" in the case of executing a command [22].

An interesting example is MyExperience [34] which collects many variables related to the use of the mobile phone. A major drawback with the approach is the use of "surveys" that are triggered by certain events. Applications that require user input will generally become uninteresting to the users over protracted periods of time.

However, prompting the user will remove most of the uncertainties associated with context-sensitive computing. Therefore, instead of guessing at the user's context and making automatic decisions, the user inputs what he or she prefers at the specific moment. In an almost identical context, the same user may still have different preferences at different moments and it makes sense to ask the users their preferences and learn from the historical experiences of user preferences. When the preferences are learnt, the system becomes better placed to make automatic judgments. These are inherent problems when dealing with context-sensitive systems, but prompting the user is not an ideal way to deal with uncertainties.

Automatic Settings

This category of systems seeks to act on a context situation without the aid of the user. The applications have the ability to intelligently interpret the user's environment and reconfigure the mobile terminal dynamically according to new context. This is a comparatively more complex approach because it involves making decisions for the patient. While this is an ideal situation, a major drawback is that the application will not always know the user preferences. User preferences may be different in nearly identical contexts.

Bellotti[39], who is strongly opposed to what Schilit[22] calls *"automatic contextual reconfiguration"* and *"context-triggered actions"* argues that the application must always warn the user what actions certain contexts trigger and advise the user of the general capability of the context-sensitive application. She further argues that the application must always ask the user whether or not to go ahead with certain actions because people's preferences change.

A major advantage of this approach, though, is that complex situations can be modelled easily without the patient being aware of the complex decisions being made. In most instances, the user is not fully aware of the implications of certain context information, especially relating to expert domains such as health care, physiological measurements or communication technology.

An example could be a mobile unit that operates in environments where there are often weak or intermittent connections. A context-sensitive mobile phone can download or pre-fetch data when it predicts a disconnection based on location and/or signal strength. The mobile phone can then disconnect and use the downloaded data and effect updates when a connection is re-established. All this is transparent to the user because waiting for user input would present a computing bottle neck and possibly defeat the purpose.

Other Similar Classifications

A similar classification was done by Mäntyjärvi [40] and is very similar to the above, albeit Mäntyjärvi adds an intermediate stage of semi-automated. Therefore, any element that would represent traits of both manual and automatic, as classified in earlier paragraphs, could easily fit into the semi-automatic category.

Another classification was done by Barkhuus [41] and includes active context-awareness, passive context-awareness and personalization. Again this classification is similar to original classification; with active context-awareness corresponding to automatic settings and passive corresponds to manual settings. Just to get a feel of the popularity of either approach, I have tabled some projects and the approach or approaches they use:

Table 2: Example projects' orientation

Project	Manual	Automatic
Stick-e [23] [32]		X
SmartReminder [33]		X
MyExperience [34]	X	
eWatch[38]		X
SenSay [35]		X
TEA*	X	X
SenseWear [36]		X
IST Vivago [37]	X	X
ContextPhone [28]	X	X

From above it would seem that research efforts so far favour automatic context configurations. This is not a surprise because "smart" in the context of ubiquitous computing generally refer to being able to do things for the patient while the presence of the underlying computers is hardly noticeable. However, there still remains some small portion that is manual to cater for uncertainties or ambiguous situations.

2.3 Core Capabilities and Design Guidelines

Classifications done by some researchers have had biases towards specific situations. Pascoe [26] describes what he hopes is a generic approach to enumerating the capabilities of context-sensitive applications. These capabilities are a form of classification and are listed below:

- Contextual Sensing
- Contextual Resource Discovery
- Contextual Adaptation
- Contextual Augmentation

The only limitation with this analysis is that it was based on wearable computers. This attempt was supposed to conform to a generic model, but fails significantly, since its application is limited to wearable computers. It does not provide capabilities that are common to all context-sensitive application on mobile platforms. However their Contextual Information Service (CIS) offers some consolation as it tries to provide a generic way of handling context information extracted in different data formats.

Jonna [42] discusses 10 design guidelines for developing context-sensitive applications. Most of them may not be critical but they help explain the risks or pitfalls of disregarding some basic design issues. One of the issues discussed is that of uncertainty of the context or situation. It is easy to make wrong inferences based on context information and it is valuable to know the probabilities of the possible scenarios.

Another important factor is the prevention of interruptions with user interactions. A lot of interruptions will become as *"spam"* and result in the user discontinuing use of the context-sensitive applications. Other considerations mentioned in the guidelines pertains privacy and the fact that the user is mobile and can only attend to one short activity at a time, otherwise we risk a technology failure discussed earlier in [16].

2.4 Context Modelling and Abstractions

The fact that context is as difficult to define as it is to model is acknowledged by many researchers [43-45] and modelling context information is a rich research area since very few researchers have successfully come up with universally acceptable models, representations or abstractions of context information. As I alluded to in previous sections, many of the context-sensitive applications are tightly coupled with certain situations and custom-built sensors, making the representation of the context information only to suit the specific situation.

It is also generally difficult to model context information. There are so many variables that influence situations and activities that the computer cannot process. This is no surprise because not even humans always correctly detect context properly and accurately. For instance, it is easy to misinterpret someone's intentions, actions or meaning [46] because one lacks all the information or one misinterprets some information. In the next sections I describe basic ways to model context information and detail some abstractions as explained in literature.

2.4.1 State Machines

Context information is a combination of different states of the variables that characterize the context. A plausible approach of using context information is modelling it using state machines or finite automata. This has been the basis of many researches in context-aware computing [35, 47-50]. Other modelling methods stem out of state machines and the use of state automata implies also the use of some rules relating to the combination of the different states.

An example project, the SenSay project, uses Moore finite state machines to model the state of the phone at any particular time. The project uses a mobile phone and a "*sensor box mounted on the hip*" [35] that communicates with the phone. Their use of state machines on the phone is shown in the diagram below taken from the same paper referenced above:

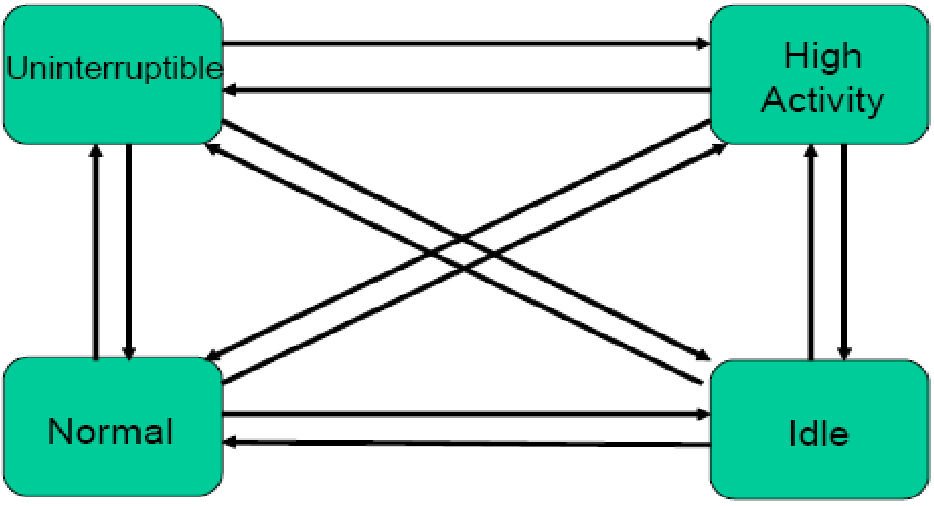


Figure 3: SenSay state machines with transitions (Source: fig.10 in [35])

The state machines are used in a limited scope to represent only the broad state of the phone. The states as shown above are the complete set and the transitions are illustrated by the directional arrows. Basically the phone can move from one state to any other. The way calls are handled will depend on the context state of the phone. Information from the sensor board is used to determine the state of the phone. The way state machines are used is very broad and not specific to the different data sources that comprise the individual states.

The use of finite state machine in most projects is theoretical and a matter of principle. Few of them, however, successfully use the machines as a basis for modelling the individual context variables or abstracting context information.

2.4.2 Deterministic and Probabilistic Models

Inherent in all context-sensitive application designs is uncertainty. Some researchers chose to deal with it by prompting the user for example while others chose to use probability of certain contexts being true. Another group of researchers totally ignores uncertainties and deal with what is obvious or given and learn from user experiences over time.

Brdiczka, et. al. [49] proposed a probabilistic approach in circumstances where the context input is highly uncertain otherwise, they say, a deterministic approach would be better. In the paper they document that both approaches can be used in real life. Their deterministic approach uses finite state machines as a subset of Petri nets and the probabilistic approach uses the Hidden Markov Models based on probabilistic finite state machines. In practice, only part of the context information can have definite interpretations, that is, we can not totally rely on all the information from different sources to be accurate.

Probabilistic approaches can not work in isolation; they have to have a real basis for deriving the probabilities. History could be a good basis for deriving probabilities or likelihood, because application users relate to past experiences [46] and a reflection of past preferences often determines current preferences. Context-sensitivity includes the users' activities in the different contexts, that is "*activity happens within a context*" [51]. Activity theory [52] describes how "outcomes" are generated from user activities and in the context. The fact that user activity affects the outcomes underlines the need for monitoring such activity over time. The likelihood of actions can then be used as probabilities when the context is not so clear.

An interesting research came up with context modelling using a *"transition probability graph"* [36]. This approach is based on learning algorithms built on user experience. It is applicable to wearable computer systems, where the system eventually learns user preferences with time. However, the way the authors evaluate usefulness and accuracy of the system leaves a lot to be desired. The research uses two young graduates in their 20's to annotate their context and compare to what the system has already learnt. After a couple of days, the researchers conclude the system was useful and accurate, but the credibility of such results is questionable to me.

Deterministic approaches are quite naive because they assume the context detection is accurate and thus always have a deterministic transition function. The probabilities of whether such a transition is likely are totally ignored, the approach is used for its simplicity. In most circumstances deterministic approaches are sufficient and are all we need because the complexity of probabilistic approaches do not give a total guarantee of accurate context detection anyway.

2.4.3 Other Modelling Approaches

Another research came up with *"context-sensitive data structures"* (CSDS), where the authors propose priority queues, lists and trees [53] as abstractions. The aim was to offer an abstraction so that the programmer would not need to worry about the context information itself or content. This research does not even hint how the context information in the proposed entity is represented. However, the research does prescribe an architecture, which includes some middleware for handling context information items.

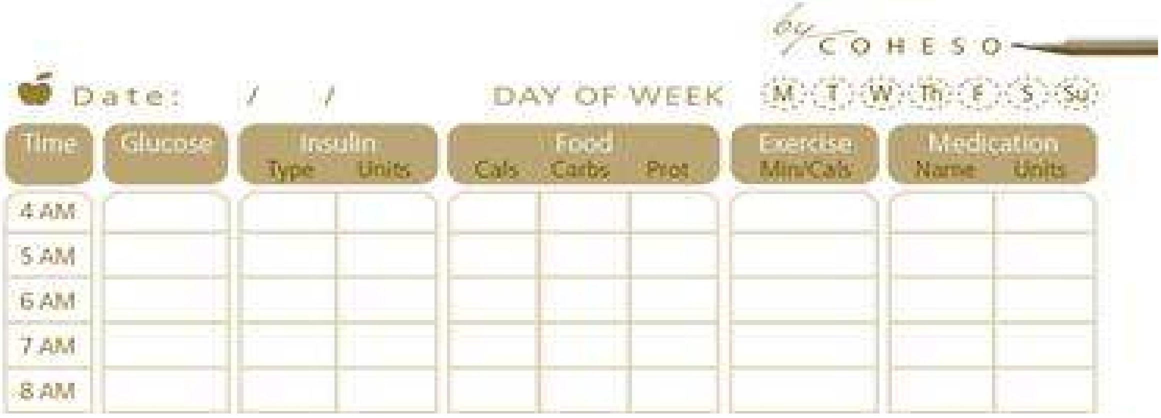
The approach taken by Flora [54] prescribes an infrastructure that tries to provide "*uniform context abstractions"* for all context information sources. The infrastructure looks convincing but the details of the context ontology that the authors propose are not given. Therefore, they only provide an idealistic high level conceptual model that can be difficult to implement. A model is needed to represent context information in a generic way, despite the heterogeneous data sources, and this is one of the major objectives of this thesis.

2.5 Type 2 Diabetes Management

2.5.1 Traditional Manual Approach

There are two main variables in addition to insulin that directly affect the blood glucose levels in Type 2 diabetes patients, and they are dietary habits and physical exercise. Intake of high energy food without a corresponding increase in physical activity can result in a too high blood glucose level and subsequent medical complications. The role that a patient plays in the care process is monitoring the blood glucose levels by taking measurements at least once a day. The patient is supposed to achieve and maintain a balance that will result in healthy blood glucose levels.

Traditionally, patients record such measurements in a paper diary and make analysis on the manually recorded data. With the information age, this has become unacceptable and obsolete although still in wide use today. An example paper diary, top headings, by Coheso [55] is shown in the figure below:



by COHESO

Date: / / DAY OF WEEK M T W Th F S Su

Time	Glucose	Insulin		Food			Exercise	Medication	
		Type	Units	Cals	Carbs	Prot	Min/Cals	Name	Units
4 AM									
5 AM									
6 AM									
7 AM									
8 AM									

Figure 4: Example of Diabetes Management Paper Diary (Source: [55])

The above diagram assumes the patient is taking insulin as medication. In many Type 2 diabetes instances, the patient does not take any insulin, especially at the early stages of the disease. The critical need to closely watch diet, blood glucose levels and physical exercise compels the patient to adopt a healthy lifestyle. It still takes some effort to access, aggregate and analyse the data read from paper diaries, hence the need for electronic diaries. Because of the commercial success, several vendors already market electronic diaries. Most of the diaries remain tied to the vendor's web servers and patient information is accessible using the vendor's secure protocols[2][3].

The limitations that paper diaries have, has lead to emergence of telemedicine tools which have been used to monitor the patient and acquiring patient data remotely. Most of such tools use a web portal to deposit patient data, making the data repository available to authorised people such as the physician or, in the instance of children with the disease, the parents. Research has proven the efficacy of such tools especially with respect to clinical outcomes[56] [57]. It remains a challenge proving the economic benefits therewith because there have not been any large scale deployment of such services at communal or national level.

Further technological improvements have made mobile phones available at low costs and with improved processing power and bandwidth. This has pushed the research community to tap into this resource and focus on ways to harness the mobile technology. In the next section I discuss research directions related to mobile applications in chronic disease management generally.

[2] http://www.homecarecenter.com/Diabetes/English/help/desktop.asp

[3] https://www.betterdiabetes.com/Info/AboutBDOutside.html

2.5.2 Modern Mobile-Based Chronic Disease Management

Many emerging disease management tools have been web-based [58] [12, 59-61], where the idea is to create a repository that can be accessed easily and securely by authorised personnel. A common model includes a mobile phone as an intermediary between different kind of sensors and the web server. The patient's physiological data is transferred to a web-server via a mobile phone. In most instances, the processing of patient data is done at the remote web server because the mobile phone has limited processing capacity. The processing can also culminate in specific values being incorporated in an EHR [62].

Today trends are changing to the mobile phone because its convenience has already gained wide acceptance. Initially, the mobile phone was supposed to improve the patient's compliance with the web-based system, but now researchers are working towards improving the computing capacity of these phones while minimizing costs. The increasing computing power and security makes it possible to easily incorporate healthcare applications into these small devices.

The use of mobile handheld devices in healthcare is increasing. Mobile terminals are now in use in managing chronic [63] conditions such as diabetes [64], asthma [65, 66], congestive heart diseases [67] and epilepsy [68]. In epilepsy, for instance, data on muscle movement is gathered from sensors worn by the patient. When a seizure is determined to be imminent, the patient can be advised to stop critical activities such as driving or using a knife. At the same time an alert is sent to the healthcare provider who promptly locates the patient.

Such proactive applications have a lot of advantages which have not been fully exploited. However, there are some limitations that hinder full adoption of the applications. For example in wearable systems, the sensors the patient must wear or use may be uncomfortable or obtrusive and the application might interfere with other applications on the phone. In other instances, the application's user interface is difficult for the patient to interact with and some people do not fancy the idea of a small screen.

2.6 Type 2 Diabetes Management - Key Result Areas

There are basically four variables that are important when self-managing Type 2 diabetes. These four variables have been clinically determined to be important for SMBG in Type 2 diabetes patients, although other variables such as weight could be added because it is a significant risk factor:

- Blood glucose level
- Nutrition/diet
- Physical exercise
- Education

Blood Glucose Levels

Blood glucose level is typically measured at least once a day, but it may also be important to measure as one feels like it. Maintaining healthy levels will prevent sickness and medical complications. Blood glucose digital meters, also called glucometers, are now a common tool among diabetes patients because of their commercial success. Glucometers are electronic devices for measuring blood glucose levels. Typically, the device comes with disposable paper strips where small blood samples can be deposited and analyzed. The main performance requirements have been for the glucometer to require as little blood samples as possible and to take as little time as possible to output the values. A glucometer is shown below:

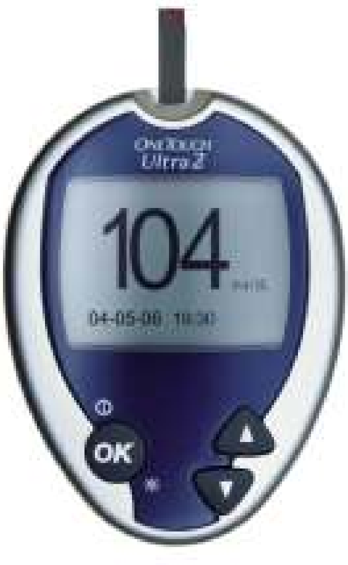


Figure 5: OneTouch Ultra2 Glucometer

The diagram above shows a glucometer with a disposable paper strip at the top. These gadgets have the capacity to store and transmit the measurements. Recent requirements include having such a device transmit to a mobile phone or desktop PC which in turn will also transmit to a web server. A glucometer used by NST has a Bluetooth adapter and can automatically and wirelessly transmit measurements to a mobile phone [3] .

Nutrition

High energy foods must have a corresponding amount of physical exercise. Diet is an important component of managing diabetes and involves monitoring the type of food one eats. A research project tried to come up with a system that automatically registers what the patient is eating. The system uses the patient's arm movements, chewing and swallowing habits to infer what the patient might be eating. Although the authors registered some success, the system's scope is limited [69].

A more practical approach was used at NST and involves a simple interface for registering food intake [5]. The system offers a brief range of food types and registration can be made about carbohydrate intake, fruit/vegetable intake, snacks intake, etc. A salient feature of the interface is the big buttons that can be pressed easily reducing major possibilities for patient input errors. Although this seems to be the best feasible solution, this manual approach probably still requires the patient to be highly motivated to comply.

Physical Activity/Exercise

Physical exercise burns up the excess energy and is therefore important for diabetes patients. There are quite a lot of systems used to measure the patient's physical exercise and activity in general. The systems include capabilities for measuring the patient's acceleration and direction. Some of these systems are complex with some intensive computations and are unlikely to result in major success on ordinary mobile phones. An example is body area networks and an example is given in the diagram below:

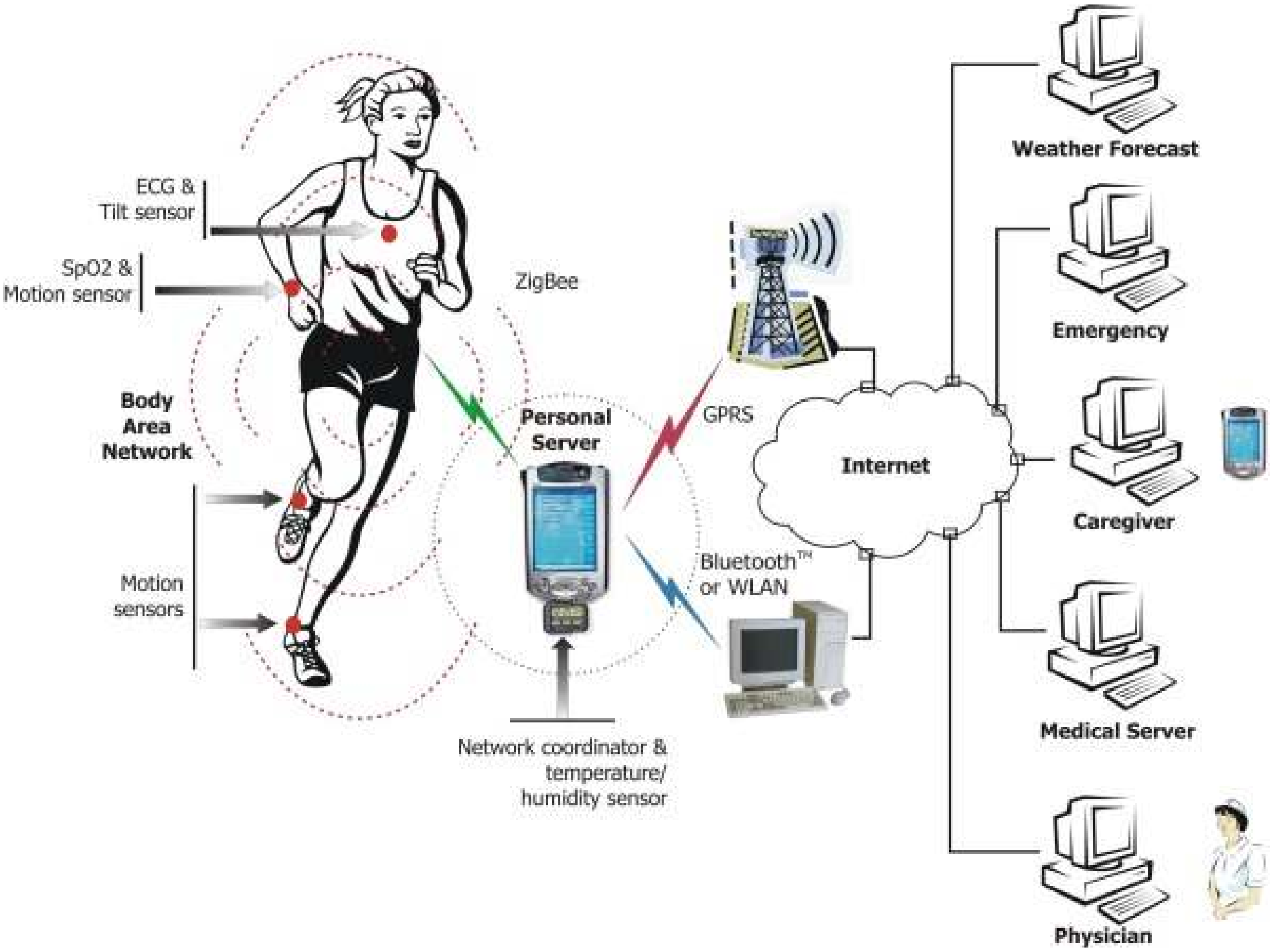


Figure 6: Body Area Network (Source: [65])

As shown in the diagram, the activity sensors are normally wearable, hence can also be obtrusive and sometimes intrusive. The BAN can be used not only to detect activity, but also to monitor vital signs parameters. BANs are getting more sophisticated and the use of wireless networks is common. The long tern medical implications of data transmission frequencies must be taken into account since the sensors are normally attached to the skin.

An approach that is becoming popular is the use of a step counter [4]. A step counter can count the number of steps and makes it easy for the patient to set targets. This is a much simpler sensor that is now being built into mobile phones. They are used to detect how much physical activity the patient is engaging in per time periods.

Education

Research has shown very little benefit to none at all of SMBG for Type 2 diabetes patients who do not use insulin [70] [71]. Some researchers have totally denied that any benefits accrue and believe SMBG actually reduces well-being [8, 72, 73], worsen HbA_{1C} and adversely affect the patient psychologically or otherwise. The reason may be because of general patient ignorance about the disease, the many educational strategies used today. For instance, a patient who does not know that reducing high energy foods and doing more exercise is helpful when the blood glucose levels are on a high trend, is unlikely to benefit from SMBG tools.

However, this situation can change if the patient is fully empowered by enough education [9]. An educated patient will know what action to take in order to achieve good long term blood glucose levels (HbA_{1C}). Today, many diabetes management systems also involve education tools. On mobile platforms, this implies a lot of educative messages and tips arriving on the device. Several papers have been published demonstrating the feasibility of SMS as a tool for education for diabetes patients, good examples in [6, 74].

2.7 Healthcare Context-Sensitive Applications

Research on context-aware healthcare systems has not yet reached maturity despite a whole decade of concerted efforts. In spite of all the research and proposed approaches, we have not seen any particularly successful context-sensitive healthcare application design. Context-sensitivity is most useful in healthcare because of the possibility of incorporating the patient's physiological data as well as other contextual data. Thus, there is a potential to make usable and reliable disease management tools, especially for high risk patient groups. However, there has not been any agreed on format or method of modelling context information.

Until when an agreeable format can be established, it seems the research community will continue to make varying, and often incompatible, suggestions for context abstractions. The problem could be with the elaborate algorithms which are not suitable for mobile platforms, especially because of the limited battery life. In the future, such algorithms might be good, but in the mean time there is obvious need for conservative approaches.

Another possible barrier is the absence of integrated sensors. Most research projects have had to make their own sensing hardware and communication protocols. This has resulted in context-sensitive applications that are closely tied to the custom built sensor system. Lack of standardization will delay context-sensitive systems to come into wide use.

There has not been any serious evaluation of healthcare context-sensitive systems. Little information is readily available on the usability of such systems. For instance, how users have perceived the systems and whether users think it is a great idea. One might want to know how successful the context-sensitive systems have been so far. What would be the consequences of wrong context interpretations, especially the clinical implications? Do they actually enhance quality of disease management in healthcare? It is clear that additional work is needed before we come to a complete understanding of why context-sensitive systems, especially in healthcare, have not been widely adopted.

2.8 Summary

The chapter has described an overview of the different approaches used to construct context-sensitive applications. Examples have been given of different projects and the approaches they took and classifications were made. In addition to classifying context information, a broad classification of context-sensitive applications was made as being driven by:

- Manual Settings
- Automatic Settings

An important matter relating to the different approaches for modelling context information was covered and explained why this may be the most important problem when constructing context-sensitive systems. The chapter also described how most projects base their implementations on the notion of finite state automata, albeit at a broad and very high level of abstraction.

Further analysis of context modelling approaches regarding uncertainties was resulted deterministic and probabilistic approaches. Both approaches are in use, but deterministic approaches are easier to implement while the probabilistic approach uses highly subjective probabilities.

The last sections describe the important aspects when self- managing Type 2 diabetes and the aspects affect interaction with the patient and are listed as follows:

- Frequent blood glucose measurements
- Nutrition management
- Management of physical exercise
- Sufficient education

In the end, the chapter gives a general overview of healthcare context-aware applications and concludes that such applications are few and not in wide use today. Past attempts at modelling context have been tightly coupled with the custom-built sensor boards, hence the general lack of universally accepted or generalized models. Therefore, the area is a highly significant field for further investigation.

CHAPTER 3

Methods

3.1 Research Paradigm and Tools

The thesis uses the engineering approach as described by Denning [17] in a task force committee report titled *"Computing as a Discipline*". He describes several design paradigms, but the engineering approach as described in the paper has the following aspects, which I follow:

(1) State requirements;

(2) State specifications;

(3) Design and implement the system;

(4) Test the system.

The process is iterative because often the system tests may reveal more requirements or require that the original requirements be adjusted. Therefore the process of refining will result in a better product. In contrast, the Waterfall model prescribes that each stage must be perfected before moving on to the next. This approach is impractical for many complex projects, hence my preference for the alternative iterative approach. I constructed a prototype to demonstrate my findings and the development tools I used are listed below:

Device	HTC Touch mobile phone (P3450)
Languages	VC++, VC#, XML, UML
IDE	Visual Studio 2005
SDK/OS	Windows Mobile 6
Major Libraries	.NET Compact Framework 2 SP 2
	Microsoft's P/Invoke Services

3.2 Data Collection & Experiment Methods

There are two main types of data that I use to evaluate the success or failure of the prototype. The first directly relates to the functionality of the context-sensitive framework while the second relates to the mobile phone's resource utilization and integration with the normal phone communication functions such as SMS/MMS and calls.

A. Functionality

The primary experimental method is controlled experiments where I simulate many different possible contexts and note the behaviour of the phone. Initially, however, the framework is installed for two people, to use the system over a one week period. The people evaluate the system and give me useful feedback. The iterative process allows taking into account their suggestions when the initial prototype continues to be evolved. There are some critical areas that must be analysed after data is collected concerning the following issues:

Interruptibility, Obtrusiveness and Transparency
The system is supposed to be hidden from the user and it is important to collect data concerning interruptibility. The system is supposed to reduce interruptions and the experiments take note of incidences where the user is disrupted from critical activities. It can be tricky using the user's *"response time"* as a measure of effectiveness as discussed in [27]. The patient might respond quickly because the phone has mad an inappropriate behaviour and the patients wants to quickly stop it. Alternatively, a longer response time might mean the alert has not managed to catch the attention of the patient.

For experimental purposes, the users will click a button when the phone misbehaves. The button will capture the state of all context variables at that moment and log them for review. Transparency is important because the framework must work without the patient necessarily aware of the framework's presence. Therefore, most decisions are made automatically without user input. There is a risk of wrongly interpreting the context and consequently behaving in an unexpected manner. Such misbehaviour is interesting to collect information on and will be used to evolve the prototype.

Glycaemic Control
This research does not aim to measure the clinical benefits of a diabetes management self-help tool. The benefits have already been demonstrated through several researches both at NST [5]and elsewhere [9] [7, 64]. I do not take any data concerning whether or not there was an improvement in the long term blood glucose levels.

The main aim of the research is to increase usability so that the patient can continue to use the self-help tools into the long run. It is then hoped the continual use of the self-help tool for a reasonable amount of time will help the patient maintain a healthy blood glucose level.

B. Resource Consumption

This data collection is about the performance of the framework on the mobile phone. I take note of some important variables after installing the system and use a mobile phone without the system as a control. The main variables I am interested in are:

- Main Memory Use
- Battery Consumption
- CPU use

Main Memory Use

Main memory is crucial to the proper functioning of the mobile phone. If there no longer is any memory, the device will not launch new applications unless some applications are closed or memory is freed. This is a major drawback of mobile platforms because they already have limited memory. The prototype is expected to use as little memory as possible. Data is collected on memory usage and used to evolve the prototype. Secondary storage is becoming less important to consider every day because we now have large amounts of secondary storage with fast access. Data will be collected on the exact memory usage as a percentage of the total memory available.

Battery Consumption

This resource is also very crucial and is a huge limiting factor for many potential mobile applications. Many applications have not been successful on mobile platforms because of their huge battery consumption requirements. Data will be collected on how the prototype impacts on the battery life. Average battery life after the prototype is installed will be compared against a control mobile phone.

CPU Use

The use of CPU time is important because computer users demand short processing times and faster programs. Data about the effect of the prototype on processing speed will be used and also gauged from user comments. In practice, the speed will depend on many other factors such as the number of installed and running applications.

3.3 Evaluation Methods

It is important to get general feedback on how the system appeals and about the functionality; what should be added, removed or improved. This part has a subtle complication because the prototype does not have any visible user interface. Therefore, the behaviour of the mobile phone is assumed to be influenced solely by the prototype.

Success is rated by a high ratio of correct context interpretation against false interpretation. The prototype must exhibit few cases of misbehaviour or inappropriate reactions. Analyses are done on the data collected by the prototype itself.

As explained in the preceding section, the prototype for this experiment will have a button that the test users can click to indicate misbehaviour. This button triggers capturing of the state of all context variables. Depending on the state of the context captured, I will be able to compare with the prototypes interpretation of the situation. Such information will be used to make the prototype adapt to different patient preferences in the evolving prototype.

3.4 Critique of the Methods Used

Using controlled experiments is a good way to test the robustness of a prototype. It is easier to test the prototype for extreme conditions and it is easy to subject the prototype to all the possible situations. This would be impossible if a field test were to be done over a short period such as the one for which this research was. However, controlled experiments tend to give biased results. This is because I carefully select the situations and gave my own opinion on how I think the prototype has behaved. The frame of reference is my knowledge, preferences and experiences, which may differ significantly with some of the potential users.

Field tests with diabetes patients would probably yield more practical results. However, the disadvantage is that we may never know how the framework behaves in rare contexts or circumstances that the sampled patients do not incur during the short experiment phase. In addition, in Norway, legal restrictions and bureaucracy may have resulted in heavy delays. For proving my concepts and the claims that I make about the framework and prototype, controlled laboratory experiments are sufficient.

3.5 Summary

This chapter discussed the methods used for the following:

- Prototype design (Engineering approach)
- Data collection (Small survey on 11 diabetes patients)
- Experimentation (Controlled experiments and colleagues testing)
- Evaluation (Accurate context detection and efficient resource utilization)

The chapter describes how the thesis follows an engineering design paradigm for developing the prototype, coupled with an iterative approach. The methods used are meant to be able to ultimately evaluate the feasibility of the framework in terms of functionality and usability, and how the framework affects the limited resources on the mobile phone. The chapter also explains why an iterative approach was necessary and concludes with a critique of the methods, clearly explaining why controlled experiments yielded superior results under the circumstances.

CHAPTER 4

Requirements specification

It is important to state the two main and broad requirements at this point:

- Accurate detection of contexts
- Minimal resource utilization

The thesis uses the Volere Requirements Specification Template [75] as the basis for most of the analysis in this chapter. There are certain assumptions made about the prototype regarding the already existing components and data availability.

I. It is assumed most of the data is available on the terminal's memory (secondary storage). For instance, physical exercise and activity will be measured by step counter data. NST has a prototype that wirelessly transmit data from a custom built step counter [4]. Data that cannot be automatically transferred from the sensor will be modelled and simulated.

II. Another assumption is that the terminal is operated by one person, who is the owner and therefore, sensitive information about the health status of the user can be communicated without authentication. Otherwise the usability may decrease if the user is authenticated each time before viewing messages or historical information.

III. It is also assumed an integrated set of sensors is available on the phone. For example, basic camera should be part of the phone.

4.1 Sources of the Requirements

The main source of the requirements is based on previous work done in the "Lifestyle" research group at NST. The group has been working on the diabetes self-help tool [2] and recruited 14 diabetes patients. Their early work has therefore been influenced by real potential users and some of their requirements have resulted in the need for this thesis.

Other requirements are taken from advice given by specialists and consultants conversant with self-management of diabetes. The iterative process continues to yield more requirements, and probably demand adjustment of the current requirements as well.

Other requirements are also taken from the survey conducted on 11 of the recruited diabetes patients and these mainly relate to ideal functionality and prioritization of context variables. The survey was a platform to give them a say in the services they would like to see implemented on their phone so that the self-help tool would be more enjoyable to use.

The target age group was established to be people past their middle ages, that is, +50 years old. The age distribution of the 11 respondents is shown in the figure below:

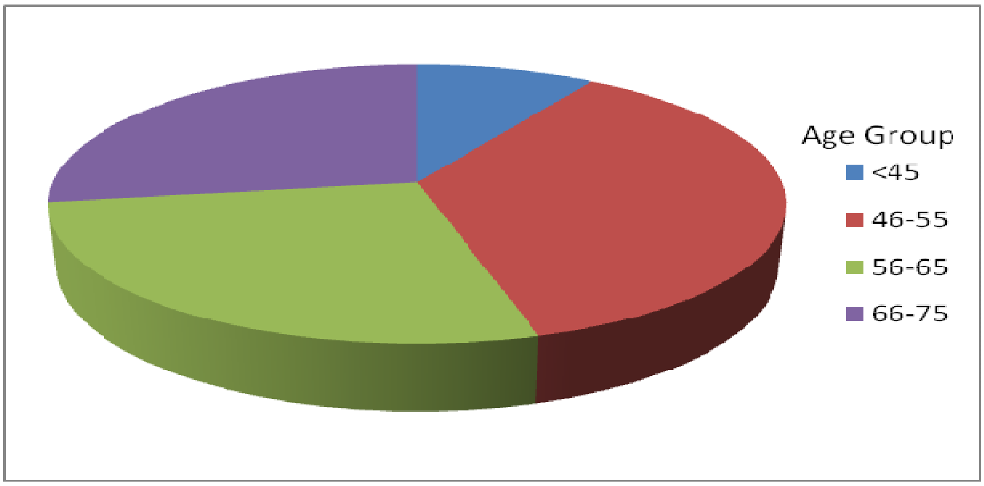


Figure 7: Age distribution of survey respondents

The average age has a lot of implications on what context information should be given priority. This is because at such ages the patients' senses and cognitive abilities diminish. Therefore, context information that has to do with ambient light, for example, should be prioritized because the patient's eye sight is most likely to be faltering. The figure below illustrates the level of literacy among the respondents:

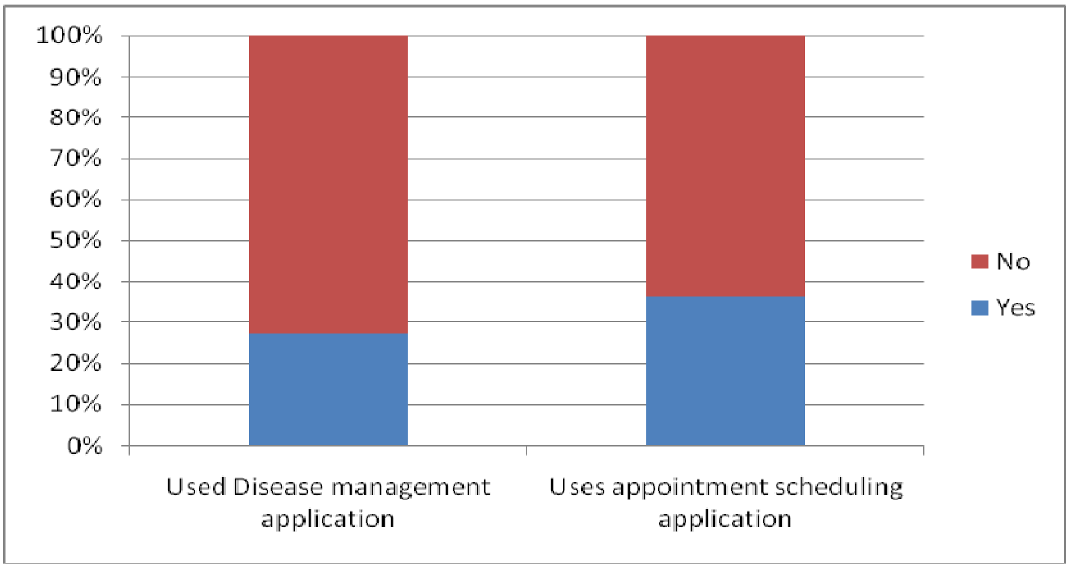


Figure 8: General computer literacy indicators

From the figure above, we can see that most have not used any electronic disease management application. Most of the respondents did not use any electronic calendars on their mobile phones. This information is useful when I shape the kind of contexts that the framework is required to detect.

4.2 Alternative and Rejected Requirements

There are several ways of approaching context-sensitivity for mobile communication and Schmidt [21] explains the complexity of context, the hierarchy of context information and how context includes many factors and variables. A sample route in the hierarchy as described in the paper is shown below:

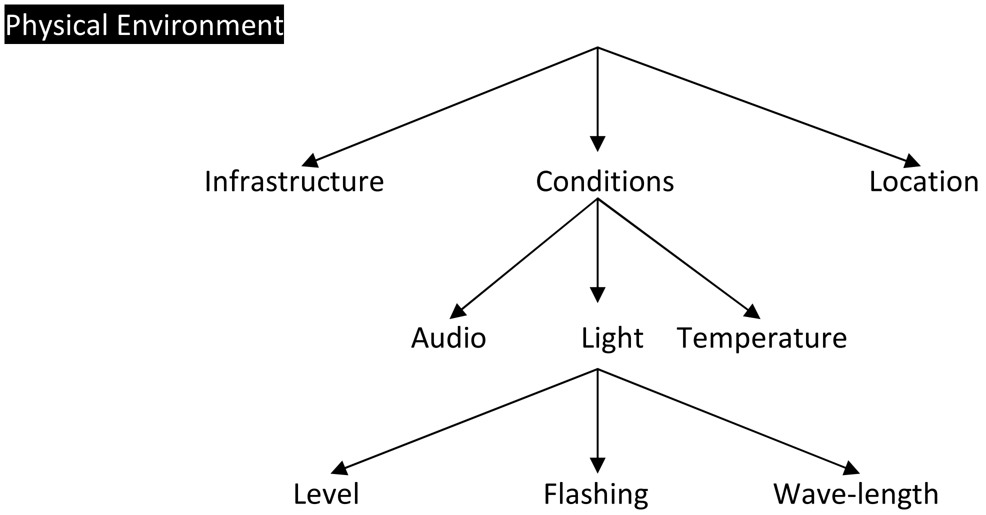


Figure 9: Context Hierarchies (adapted from Fig.1 in [21])

The figure above illustrates how a lower level context variable is an attribute of the parent, that is, the upper level context unit. Thus, besides choosing relevant context variables, the right hierarchies must also be chosen, and reducing the granularity will simplify the algorithms. For example one could use the following two designs:

A. Light || Dark

B. Level || Flashing || Wave-length || dark

C. Other compatible combinations

Clearly, the first option should be easier to implement than the second. Below I list some variables that are normally featured in context-sensitive applications as described in the preceding chapter on related literature, but that I have chosen to leave out. In addition, I explain why I choose to leave them out. Most of them would be interesting to implement at later stages though.

Spatial Information: Location

Location can pose a privacy threat because patients may be uncomfortable with systems that know their every move and location. While such a feature would be ideal for elderly patients and high risk patients, I deemed it not essential to this project, although it could easily be implemented by using WIFI connections to determine if the patient is at work, home, indoors or outdoors.

Environmental Information: Temperature

Temperature is important to Type 2 diabetes management, both atmospheric and body temperature. This functionality would require additional temperature sensors, wearable or otherwise and using external context sensors would defeat the objectives of this thesis.

Availability of Resources: Connectivity, Bandwidth, Signal Strength

The prototype does not need to communicate with external devices or web servers. The self-help tool prototype does not import, download or connect to external data sources or servers, except for the glucometer and step-counter with minimal bandwidth requirements. This simplifies the situation because communicating health related data has legal constraints, and is generally left for future implementations. In the future, however, this is an important variable to consider, especially if the self-help tool will communicate with external servers, patient data repositories or EHR. The extensible nature of the framework will allow such context information to be added easily in the future.

Identity

I have made an assumption that the patient is the sole owner of the device and no further authentication should be required. Enforcing stricter security will reduce usability and undermines the objectives of this research. The research on RFIDs and other identification technology is still immature.

Social Situation

I have also left out sniffing of the social situation because it is not significant in the scope of this research. I included detection of sound levels and can infer the presence of people or engagement in conversions through the ambient noise detection. In the future, however, such functionality is important because there will be usage scenarios where it is necessary to detect devices or persons and their presence or proximity.

Activity: Accelerometer

The sensor is now reasonably available on ordinary mobile phones, but not to a very large extend. The sensors need to become essential and the part of every phone and because of this shortfall, I had to use the available step-counter as a substitute.

4.3 Functional Requirements

This section details the required behaviour and functions of the prototype. I make use of an event list and use case diagrams to illustrate the different interaction possibilities. The context variables I have determined to be essential are listed in the succeeding paragraphs:

I. *Availability of Resources: Battery State*
Battery state is an important parameter for most mobile electronic devices because it usually is limited. It is therefore important to keep watch of this important variable. The prototype should have a little impact as possible on the battery life and advise the patient when it is running low.

II. *Environmental Information: Ambient noise*
Ambient noise is vital especially when the patient has hearing impairment or is elderly. Surrounding noise is important to consider for two main functions, firstly, audibility of the ring alert and secondly, to estimate whether the patient can concentrate on the message at the current ambient noise levels.

III. *Temporal Information: Time*
Time of the day, week, month or year is important to consider because a person's cognitive power changes from time to time for different reasons. For example, a person's cognitive power is likely to be lower just after midnight when the patient is overcome with sleep than at noon.

IV. *Scheduling: Calendar events*
Calendar events are the source of important information such as meetings and appointments. The system must be able to determine the current busy status of the patient. This variable is most important for the working class who may often attend meetings with important clients, and have to make presentations.

V. *Environmental Information: Ambient Light*
The amount of light around the patient and the terminal is important in order to tune the device's backlight. A dark environment will require the device lighting to be dim while in a bright environment, the lighting must also be high. This is especially important because the screen is small the user may need to view important graphs and data. If the contrast is not right, information and meaning might be lost, especially for patients with visual impairments. This variable emerged as the most important to the elderly patients as exposed by the conducted survey.

VI. *Activity: Physical Activity*
The system must be able to determine whether the patient is highly active or idle. This is important because this will affect how the patient receives alerts. An appropriate message and method will be used depending on whether the patient is sitting, walking, exercising or jogging.

4.2.1 Event Listing and Use Case

In order to systematically enumerate the functional requirements, I make use of an even list. The event list contains all the events that can take place in the system and the possible inputs and outputs.

Table 3: Event List

Event Name		Input/Output	Summary
1	Check Busy Status	Phone’s calendar	Check calendar to see current busy status
2	Check Ambient Noise	Sound from microphone (IN) Adjust smart phone volume (OUT)	Get ambient noise from microphone and adjust volume
3	Check Activity State	Step counter data	Check step counter data for idle or highly active states
4	Check Battery State	Phone’s battery API	Get the phone’s battery state using the API and output the level
5	Check Time State	Phone’s clock	Get the time from the phone’s onboard clock
6	Check Light State	Light intensity from camera (IN) Adjust mobile phone backlight (OUT)	Get the ambient light intensity and adjust the backlight
7	Call Alert	Incoming Call (IN) Deny/Alert Patient (OUT)	Handle an incoming call and either deny or allow patient alert
8	SMS/Message alert	Incoming SMS inbox (IN) Deny/Alert Patient (OUT)	Handle an incoming SMS and either deny or allow patient alert
9	Monitor context	Get relevant context variables states (IN) Take action as necessary (OUT)	Keep monitoring states and effect adjustments or alerts as needed by each
10	Adapt self-help tool	Get context variables states (IN) Adapt tool interaction interfaces (OUT)	Gets the current context and adjust the tool’s user interface and content

Using the event list above, a UML use case diagram is developed to define the prototype scope. The use case diagram graphically illustrates the prototype boundaries and the interactions with the patient and other systems, as shown below:

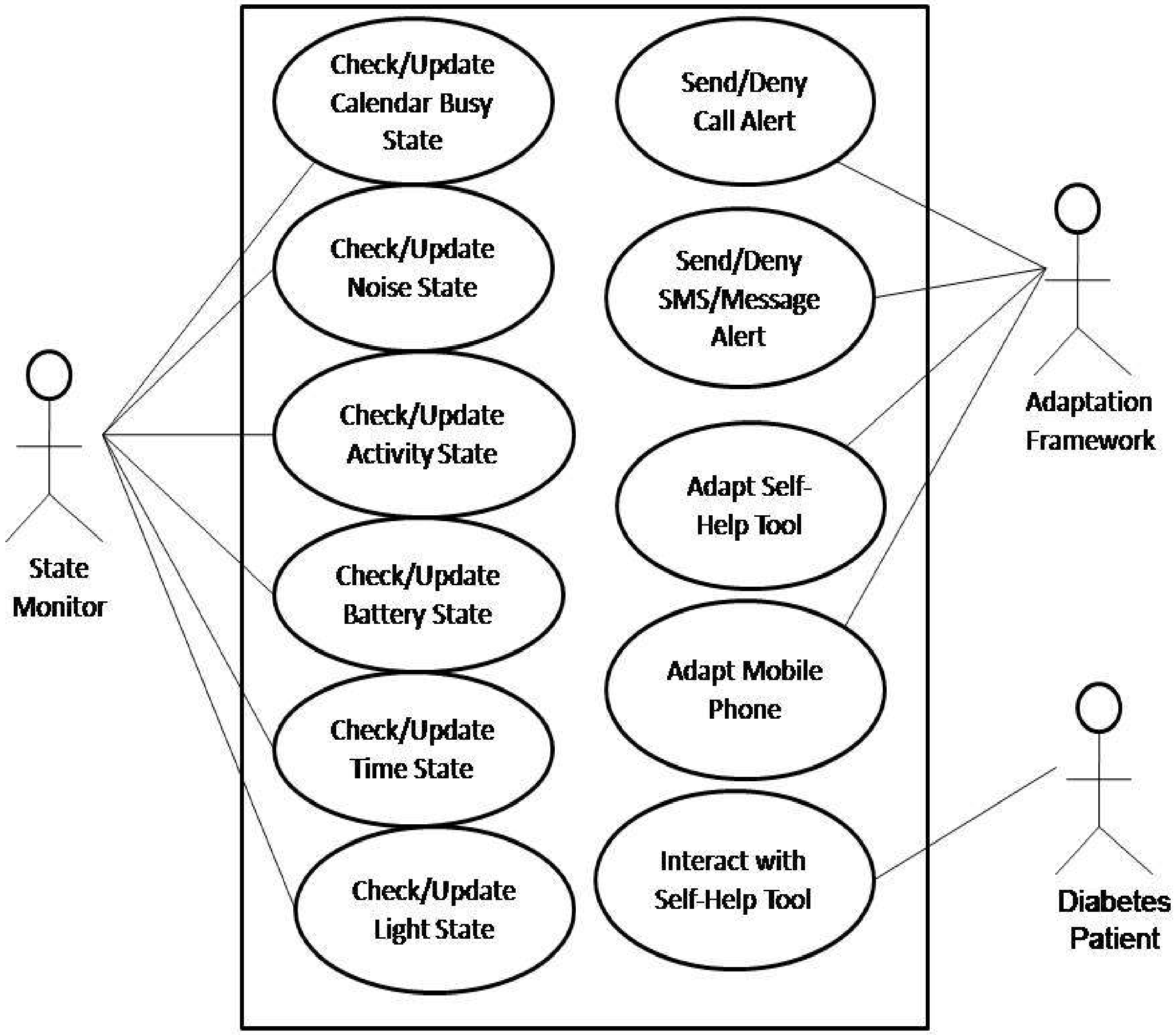


Figure 10: UML use case diagram for the context-sensitive framework

The UML diagram above is a visual aid for the functionality and boundaries of the system. However, below I further describe the events given in the table as formal and detailed descriptive requirements.

Use Case 1: Check/Update Calendar Busy State (Actor: State Monitor)
1. Read calendar
2. Check current appointment status
3. Update state machine
4. Adapt the phone alert methods

Use Case 2: Check/Update Noise State (Actors: State Monitor)
1. Read input samples from microphone
2. Store array of samples in buffer
3. Calculate average noise level from samples in the buffer
4. Update the noise state machine
5. Adapt phone volume

Use Case 3: Check/Update Activity State (Actor: State Monitor)
1. Check buffer for activity status
2. Update the activity state machine
3. Adapt the phone alert methods

Use Case 4: Check/Update Battery State (Actors: State Monitor)
1. Get battery level using .NET API
2. Update battery level state machine
3. If low and not charging then alert the Patient
4. If critical and not charging then check today's remaining appointments and alert Patient

Use Case 5: Check/Update Time State (Actor: State Monitor)
1. Get time using .NET API
2. Update the time state machine
3. Adapt the phone alert methods

Use Case 6: Check/Update Light State (Actors: State Monitor)
1. Check ambient light conditions using camera input
2. Update light state machine
3. Adapt the phone backlight

Use Case 7: Send/Deny Call Alert (Actor: Adaptation Framework)
1. Check state machines and priority levels
2. If OK then deliver incoming call alert to the Patient
3. If not OK then deny request and divert to voicemail and keep monitoring context states

Use Case 8: Send/Deny SMS/Message Alert (Actors: Adaptation Framework)
1. Check state machines and priority levels
2. If OK then deliver the alert
3. If not OK then hold message and keep monitoring the context states

Use Case 9: Monitor Context states (Actors: Adaptation Framework)

1. Check the states on the relevant state machines
2. If OK then deliver previously denied information, warning, call or message alerts
3. If not OK then keep monitoring the context states

Use Case 10: Adapt Self-Help Tool (Actors: Adaptation Framework)
1. Check the context states
2. Update the tool's graphic requirements
3. Update the tool's content
4. When tool is opened, display the tool's adapted interaction interface

* Other important variables such as backlight and volume are expected to be up to date at this point.

4.4 Non-Functional Requirements

Look and Feel

The prototype does not have any graphical user interface or requirements for user input. As such, the system is transparent to the user and the user can only see a "smart" device that does what is appropriate in any situation. Automatically reconfiguring the mobile phone will result in a truly smart application. Therefore, there should be minimal interaction with the patient. The prototype should appeal to all users of the diabetes self-help tool regardless of age. Thus, the phone should look and feel as normal, only pleasantly smarter.

Usability and Humanity

Many disease management applications are abandoned in the long run mostly because of lack of usability, for example in [16] where there was a complete failure. Developers come up with fantastic products that are, unfortunately, unusable or just not interesting enough to patients. The prototype in this research should enhance usability of the NST self-help tool on the mobile phone. The main requirement is for the prototype to correctly interpret the context and exhibit behaviour that testifies its accuracy to the satisfaction of the patient.

Performance

Performance is related to the prototype's use of the limited mobile resources. The prototype must use little resources and not make a significant change in resource consumption. The users must also perceive the phone as not to have slowed down or become less efficient in any way:

Table 4: Specific requirements for performance on mobile phone

Property	Specific Requirement
Main Memory	The allocation must not exceed 10% of RAM when framework is idle
Battery Power	Effect must be less than 20% of normal battery power consumption
CPU Time	Effect on other programs Less than 50% of their normal execution time

Security

It is assumed that the owner of the mobile phone will be in possession of the device all the time. This assumption is important because health related data is confidential, but requiring constant authentication will reduce usability. A mobile phone is generally considered private and personal because it also houses communication logs and messages of a personal nature.

Legal

There are no legal requirements intrinsic to the prototype's core functionality. However, any experiments may not be done on real patients. Another requirement is that there should not be any transmission of patient data over non-secure channels without proper authorisation. The current prototype does not in itself require transmitting patient data of any form. The prototype is not expected to be risky or harmful to potential users or patients.

4.5 Summary

This chapter describes the requirements analysis and specification based on the Volere template. The bulk of the requirements were obtained from the past research efforts at NST, from expert advice and from the pre-research survey. The first part of the chapter mentions why some of the requirements were rejected and how the rejected context variables would not add value to this thesis.

Later, the chapter articulates the accepted context variables as functional requirements using a detailed event list and a use case diagram. The chapter concludes with discussing a couple of qualitative and non-functional requirements that the prototype is supposed to meet, and why.

CHAPTER 5

Design

The thesis mainly focuses on formulating a framework for acquiring/extracting, modelling and using context information. The resulting framework presents a reusable and extensible model from which developers of context-sensitive applications can draw. Reasonable generalizations are made that apply to a broad range of situations as opposed to strict prescriptions with limited scope and application. It is important to note at this point that there has not been any universally accepted way to extract, process, represent or use context information.

There has also been more than a couple of ways of dealing with uncertainty when designing context-sensitive systems. Many research projects have been specific to certain cases or scenarios. Thus, each research has prescribed its own way of constructing context-sensitive infrastructures or services based on their experience and bias, and specific sensors. This is not surprising because context is difficult and complex.

Indications are that this is still a rich research area with potential for substantial findings in the foreseeable future. The design process in this thesis contributes to this future by proposing generic context modelling methods using generalizable abstractions.

5.1 Design Goals and Considerations

Resource Use Optimization

The major design goal is developing algorithms that optimise use of the limited resources on mobile phones. Some relevant resources to mention are battery power, main memory and processing speed. This means that as much as possible the algorithms must take into account that the mobile phone has limited resources. This is a major determinant of success and reason why most wonderful mobile applications never make it to the wider public.

Modularity, Extensibility and Reusability

Modularity is an important consideration because we have seen in the past different research projects using different context information types. Most of the context information would be dictated by the physical sensor device that the researchers build, and typically unusable elsewhere. In my design, I attach a high value of importance to modularity because it will enable components to be easily integrated and removed, hence also high extensibility. The components can also be tested independent of each other and the completeness in functionality of the individual components makes them reusable.

Fault-tolerance

A modularised system will require high fault tolerance because failure of a component can disrupt the whole system. For example if a sensor fails or for some reason is not able to acquire context information, the system is supposed to seamlessly drop the component and still work *correctly* with the other remaining components.

For the rest of this chapter I use a bottom up approach to describe design of the different relevant aspects as listed below:

I. Context extraction

II. Context modelling and representation

III. Rule design

IV. Inference engine design

V. Modelling integration with self-help tool

VI. Integration design and framework/system overview

5.2 Context Extraction

This section is a description of the design used in order to extract the context information from the different variables that were identified in the preceding chapter. It describes how context information is acquired from the different sensors and processed into usable services. Context information will be extracted from the following context variables:

1. Ambient Noise
2. Battery State
3. Ambient Light
4. Physical Activity
5. Calendar
6. Time

5.2.1 Ambient Noise

The mobile phone's microphone can be used to gather the noise level in the environment of the patient. The ambient noise level will be used in two ways:

I. To adapt the phone's volume to changing noise levels

II. To determine whether the call/message can be handled at all

For example, the patient is in a quiet place, maybe concentrating on a task. A loud ring from the phone may startle the patient and possibly disorient him or her. Similarly, a low sound will result in missed calls because the ring was not loud enough to be heard in the relatively noisy environment. Other context-sensitive output methods that can be used in conjunction are vibrations or backlight flashes.

Below I describe a step-by-step process for gathering the ambient noise data and calculating the noise level. The processes uses ideas described in [76].

1. Record short samples in .wav format from microphone
    - Using Microsoft's P/Invoke library, I capture the short sound bytes
    - Take samples for just one second at a time

2. Store the samples in a buffer
    - Create a buffer array to store the samples
    - Keep in this buffer for one minute for faster access

3. Read the samples
    - Read the buffered sound array

4. Calculate the root-mean-square (RMS) from the pulse code modulation (PCM)
    - Using the bytes stored in the buffer, calculate the average
    - This average is the estimate of the noise level

5. Return the calculated noise level and the recommended volume level

Analysing Ambient Noise Values

I prefer to use average of the sample to using a peak values for the .wav samples. The wave files are recorded into 256 Kb buffers and as much 8-bit samples as can be collected in a second. The format of the raw file is .wav in pulse code modulation (PCM) as defined by Microsoft's wave file format [77], using a single channel with a sampling rate of 11025Hz.

The actual demarcations are customisable, but for illustration purposes, I use the following initially:

Table 5: Average RMS against recommended volume levels - Example

Average RMS	Volume Level
>40	Too Loud
40 - >35	5
35 - >30	4
30 - >20	3
20 - >10	2
<10	1

From the table above, we can see that when the average RMS is bigger than a certain threshold, in this case 40, then the user should probably not be alerted because it is unlikely the patient can engage in effective interaction. The other noise levels will only result in the phone volume level being adjusted accordingly as depicted in the table.

Sensing Ambient Noise - Limitations

Detecting the amount of ambient noise using the microphone is somewhat accurate. However, determining the actual source of the noise will need additional sensors. For instance, the noise could be from a conversation, overhead aeroplane, passing vehicles or football match cheering. If I could determine that the noise is, in fact, from a conversation that the patient is part of, then the outcome would be different from if the conversion was from nearby people. I chose to ignore all these complexities and handle noise as it is. For instance, whatever the noise source, noise levels will always affect general audibility both of the ring tone and the phone conversation.

Another limitation is mapping the ambient noise levels into the 5 volume levels on a mobile phone. The loudest ambient noise permissible for any call will depend on the patient. This is a specialty field and patients may not know what is good for them. The levels I used should work for the average patient and I leave these levels customisable as needed.

5.2.2 Battery State

Power conservation is paramount for all mobile applications because battery power quickly runs out. Naturally, battery power becomes the most valuable resource a mobile phone has. It can be quite frustrating to be unable to use your device because the battery is flat and there is no electricity outlet nearby, or the charger is left elsewhere.

The battery state is easily acquired through the SystemState API available in Windows Mobile 6 SDK. It gets the battery level, status and several other variables. Below I summarize the levels available to developers:

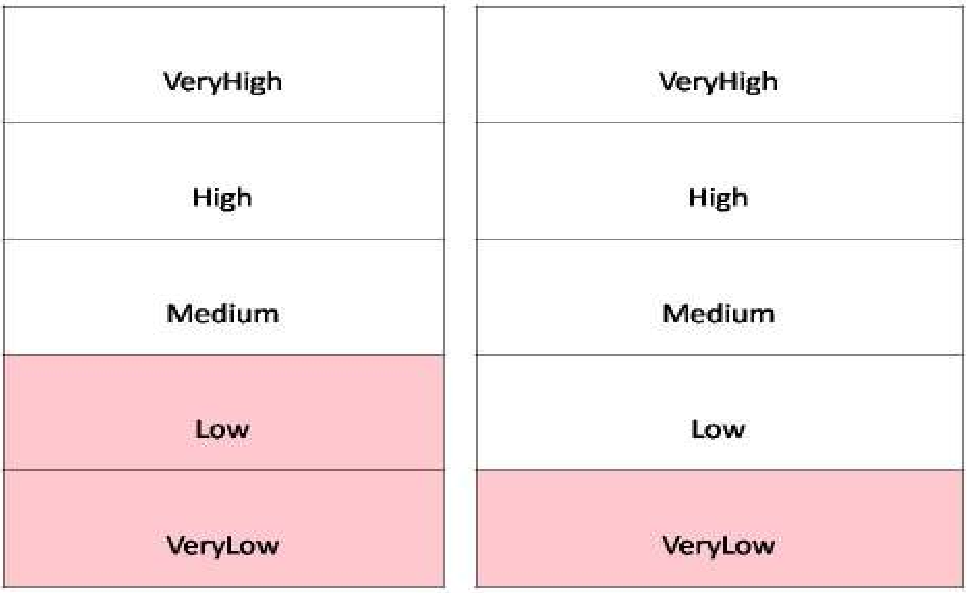


Figure 11: Battery Levels (as implemented in .NET)

From the figures above, the plain levels are good and are expected to last the whole day or until one can recharge the battery. The first scenario has the battery status “LOW” as critical, while in the other situation, the critical level is at “VERY LOW”. In this project I take the first scenario to hold but the difference is negligible, and will depend on the applications running on the phone. For example, if the battery runs out quickly, then the “LOW” level must be a critical level.

The SystemState API also provides some methods for checking another set of variables for the mobile phone battery status. Therefore the system checks whether or not the phone is already being recharged before issuing an alert for critical battery level. These statuses are summarized in the table below:

Table 6: Battery Statuses (.NET)

Normal
Not Present
Charging
Low
Critical

Such a combination is particularly useful, because a VeryLow battery could already be on recharge. Before making any conclusions, we check the current state of the battery.

Limitations

This functionality may prove redundant for the average user because mobile phone users have now gotten used to keeping their mobiles fully charged. During the early years of personal mobile phones, this functionality would have been more valuable. Today, users have learnt the consequences of running out of battery power. However, for the target group of elderly patients the functionality is deemed useful, especially in cases of mild dementia. As a function in an integrated tool that makes the mobile phone smarter, it is absolutely essential to have the battery levels kept in check.

5.2.3 Ambient Light

Estimating the ambient light without the presence of a dedicated light sensor can be cumbersome. At the moment only a handful of smart phones come with an inbuilt light sensor. Previously, light sensors have been custom built by specific research projects and only thought of as experimental.

Most mobile phones, however, now come with inbuilt cameras, but still there are no APIs that allow the developer access to camera controls. A feasible option is to use the photo's colour bits to estimate the ambient light. This option is worthwhile because it is compatible with the millions of mobile phones in use today. The procedure I use to extract this information is described below:

I. Get a sample photo in jpeg format

II. Convert picture to grey scale

II. Calculate the brightness using RGB values in random pixels

III. Return the brightness value, which is an estimate of ambient light level

Getting the image

Getting a photo automatically on mobile phones is challenging and can only be achieved by using unmanaged code and going a bit further down to a lower level programming platform. Using C++ to get the actual photos automatically is about the only choice I had. The routines I developed in C++ are grouped in a small library accessed in a dynamic link library (`context.dll`) file. This was a technique I used since my main development platform is .NET C# and did not have the functionality. For faster computation, the image is captured in `120 X 160` pixels, which is the smallest resolution the HTC Touch camera has.

Image processing

The cube below is taken from [78] and is the basis of the calculations:

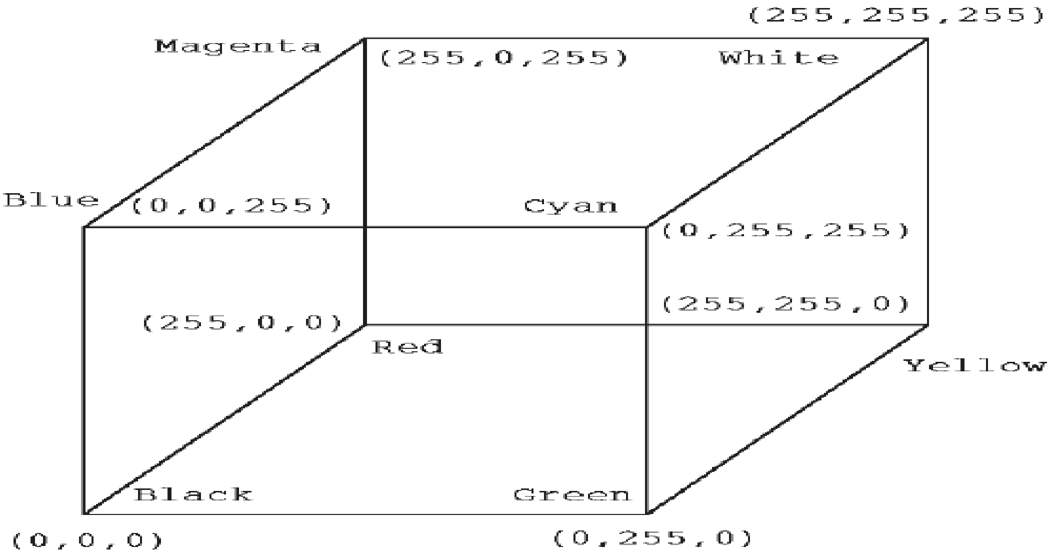


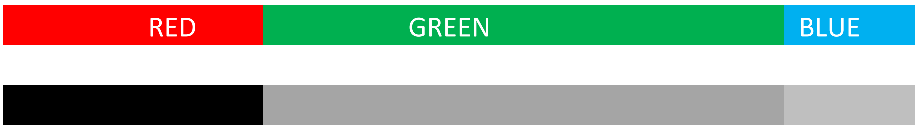
Figure 12: RGB Colour Cube (Source: Fig.1 in [78])

The colour cube above shows two extreme colours; black and white. The corresponding colours, as depicted in the diagram, are represented as follows:

```
Black:     Red   = 0                White:     Red   = 255
           Green= 0                            Green= 255
           Blue  = 0                           Blue  = 255
```

The original image is converted to greyscale in order to more accurately estimate the image brightness. The image brightness is used as the estimate of the ambient light level. A conventional formula for converting each pixel to greyscale is given as follows [79]:

```
0.299 * red + 0.587 * green + 0.114 * blue
```



The above formula will yield a value between 0 and 255, where 0 is total darkness and 255 is full light. It is compute intensive to calculate the pixels for the whole image even with the smallest resolution. Therefore, I resolved to only taking samples within the whole picture. I randomly sample four areas of 2 X 2 pixels each and average the results. This significantly reduces computation time. This technique is illustrated in the image below:

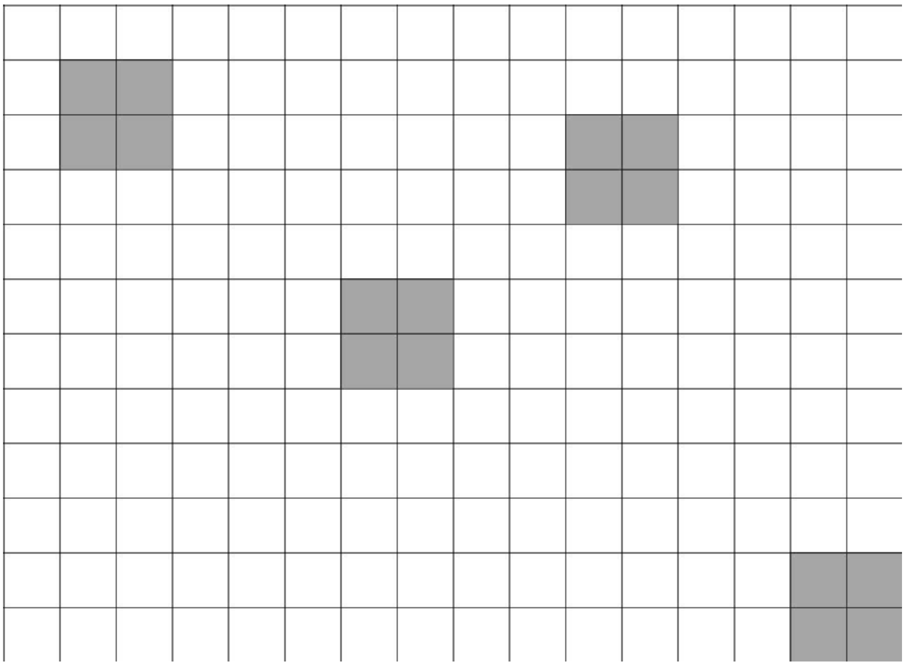


Figure 13: Scaled sample pixel map

The picture shown above represents a scaled down image but the grey pixels are the actual numbers. This technique reduces computation time by a wide margin and yet accuracy is maintained. This may be because the ultimate brightness level ranges are wide and thus strict values are outweighed by savings in computation time. This approach is consistent with the design consideration for scarce resources on mobile platforms.

Limitations

Using the camera as a light sensor is a good solution but it also has some pitfalls. For mobile phones that have camera at the back, this solution will wrongly interpret the environment as dark if the patient places a finger over the camera lenses or if the phone is laid down on a table with a dark surface. The solution works better with cameras that are fitted on the top, that is, the ones meant for video calls.

This crude method works quite well, but is not specialised and will therefore not be able to detect light sources. For instance, only light intensities can be estimated, but not whether or not the light source is a fluorescent lamp, natural light or otherwise.

Another limitation is mapping the ambient light intensity to the 4 backlight levels available on the average phone. The actual level suited to each individual at different light intensities is a specialty task, but I use settings that I see as fit for the average user and leave the functionality customizable.

5.2.4 Activity Sensing

Activity is a measurement of the level of activity at any given time. I have made use of the already developed concept of a step counter [4]. However, more functionality has to be added to the step counter. In addition to recording total physical activity for the day, the step counter is able to detect the activity state of the patient.

If the number of steps exceed 20 within a minute, then it is most likely the patient is highly active at that time. The step counter sends data to the mobile phone regarding the current activity state. When there is no significant activity, then the step counter remains dormant.

The more frequent the step counter sends data, the less the battery will last. Therefore, it will be important to establish an optimum algorithm to optimise battery life, both for the step counter and the phone. This functionality will be simulated since changing the functionality of the current step counter may take some time.

Limitations

The step counter can only detect limited activity and additional activity sensing will require more specialised sensors. Activity sensing is still a field under research and different architectures have been used to detect user activity. For example, body area networks (BAN) [80] and other variants that try and detect user activity.

Using the step counter as an alternative or substitute, however, contributes significantly to detecting user activity, albeit in a limited scope. For example, I can tell if the patient is stationery (sitting or standing), walking or moving fast. The details of user activities are, in most situations, just an over-kill. For example, when using the self-help tool, it is unnecessary to document whether the patient is turning his or her head to the left or right.

5.2.5 Calendar

The calendar is a simple source of contextual information such as whether the patient is currently occupied or not. To make it more meaningful, a priority field is attached to each scheduled appointment or task. This priority will determine of the patient should receive alerts while the appointment is running. A highly prioritised appointment will mean any calls or messages are simply not delivered until such appointment is over. Calls may be diverted to voice mail while messages alerts can be deferred. The caller can optionally be advised of the situation.

The system checks whether the patient is currently busy. Important to check is whether the patient will be in an appointment in a few minute's time. This is especially important if the messages or calls concern disease management issues. The patient is likely to be preparing to engage in the upcoming appointment.

The calendar is used as a source of important engagements information. The calendar is also used to gather the day's remaining reminders in the event that the phone's battery has run out and there is no power outlet within timely reach. The type of calendar event that is of interest is appointments, whether they are in progress, due soon or ending soon. Below I list the steps I use for extracting this information:
I. Open the default calendar in use
II. Select the appointments for the day
III. Check if any is or will be in progress soon
IV. If the battery is running out, collect all the appointments remaining for the day

The calendar routines are able to return values corresponding to whether an appointment is in progress or is five minutes to its beginning or ending. An important return value is the priority of the appointment which implies whether the patient can be disturbed or not. This priority is compared with the priority of the incoming message or call and a compromise is struck.

Limitations

Electronic calendars are a source of important information, but they can only use the information they get from patients. Patients must manually schedule events, appointments and meetings. The functionality will not work as desired if the patients do not input the schedules in the electronic diaries.

For patients who have an active formal working life, the use of electronic calendars is inevitable. This does not apply to the elderly patients who have retired or are past their normal hectic working life. The calendar functionality is always good to have in order for the mobile phone to be "smart" in a meaningful way. The survey conducted showed that only a minority of the patients used an electronic calendar.

5.2.6 Time

Time is an important context variable that is easy to acquire and manipulate. This is because of the ease of access to the onboard clock. Using .NET compact framework object `DateTime`, all the forms of dates and times can easily be accessed and manipulated.

The more used time value is the time of day because it changes more rapidly than day of week or month of year. The return values for time are as simple as invoking .Net's `DateTime` object and specifying the desired time attribute.

Limitations

The use of time is more useful in the odd hours of the day. However, one might argue that a message or call that comes at an odd hour must be urgent. True, but one could also have strayed calls from strangers or calls from an acquaintance but not intended for you. For example, similar surnames can be placed next to each other in the phone book and it is also easy to misdial using the keypad. Ultimately, it will depend on the patient, whether or not they care about calls at odd hours and the survey showed they did.

5.3 Context Data Representation and Modelling

At this point it is clear how each context variable can be represented by a number of states. Dealing with an infinite number of states from continuous data is complex and hard to model. In contrast, converting the data to discrete values, grouping the data into classes and establishing a few finite states can be a plausible solution.

Since context information is not always discrete, it is important to describe a simple process to “discretize” continuous data. Reasonable ranges must be established in order to convert any continuous data to discrete and distinguishable exclusive classes. Simple discretization removes heavy computational requirements and results in finite context states that are meaningful. The figure below demonstrates the discretization process:

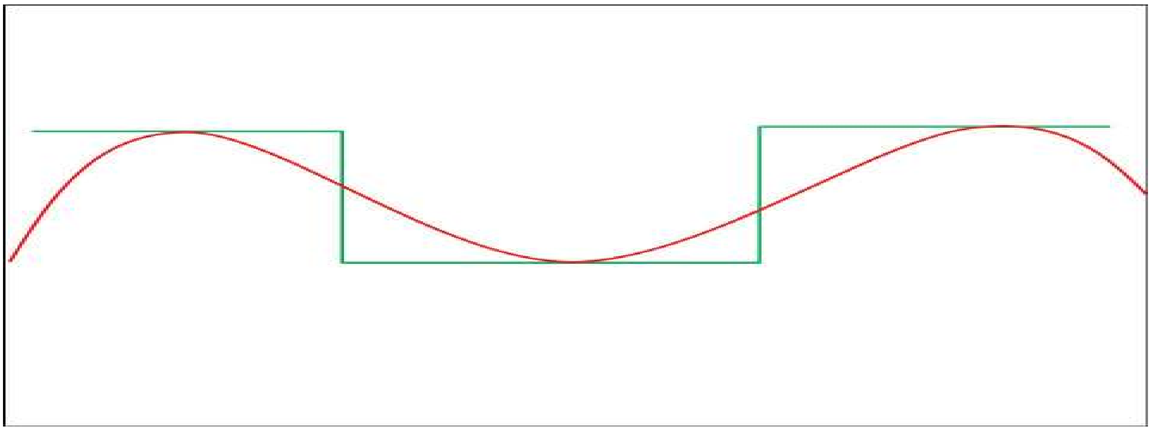


Figure 14: Discretization

In the figure above, the red line represents continuous context information values, while the green represents clearly graduated values. In the example above, the green line only takes the extreme values in the range. This is achieved in a series of initial computational steps in order to reduce the complexity of the final computations. To further explain this technique I give a sample analysis below:

```
Continuous
1st Series: 0.01     0.02       0.03       0.04 … 10.00    …     N

Discrete
2nd Series: 1        2          3          4    …     10   …     N

3rd Series: (1 - <4)      (4> - <8)       (…)          ((N-4)  -  N)
```

In the analysis above, the values could represent the initial average RMS floating point values from the sound samples as shown in the first series. The continuous data is transformed by merely rounding off the values in the first series. Finally, the sound data is grouped into discrete class intervals of approximately four members each. On a device that has five volume levels, the data can be grouped such that the classes will correspond to the number of volume levels. For this technique to work all the context information must be expressed as discrete values corresponding to the different states as described in the next section.

5.3.1 Finite State Automata (FSM)

The preceding section dealt with reformatting data into discrete values that can easily be grouped or classified. In this section I introduce the use of simplified state machines as the primary object of abstraction. The use of these simplified state machines offers a useful abstraction that removes the problem of heterogeneous data formats. In the diagram below I illustrate the simple mechanism that shifts the problem to dealing with state machines logic operations rather than prescribing methods for dealing with the different data formats:

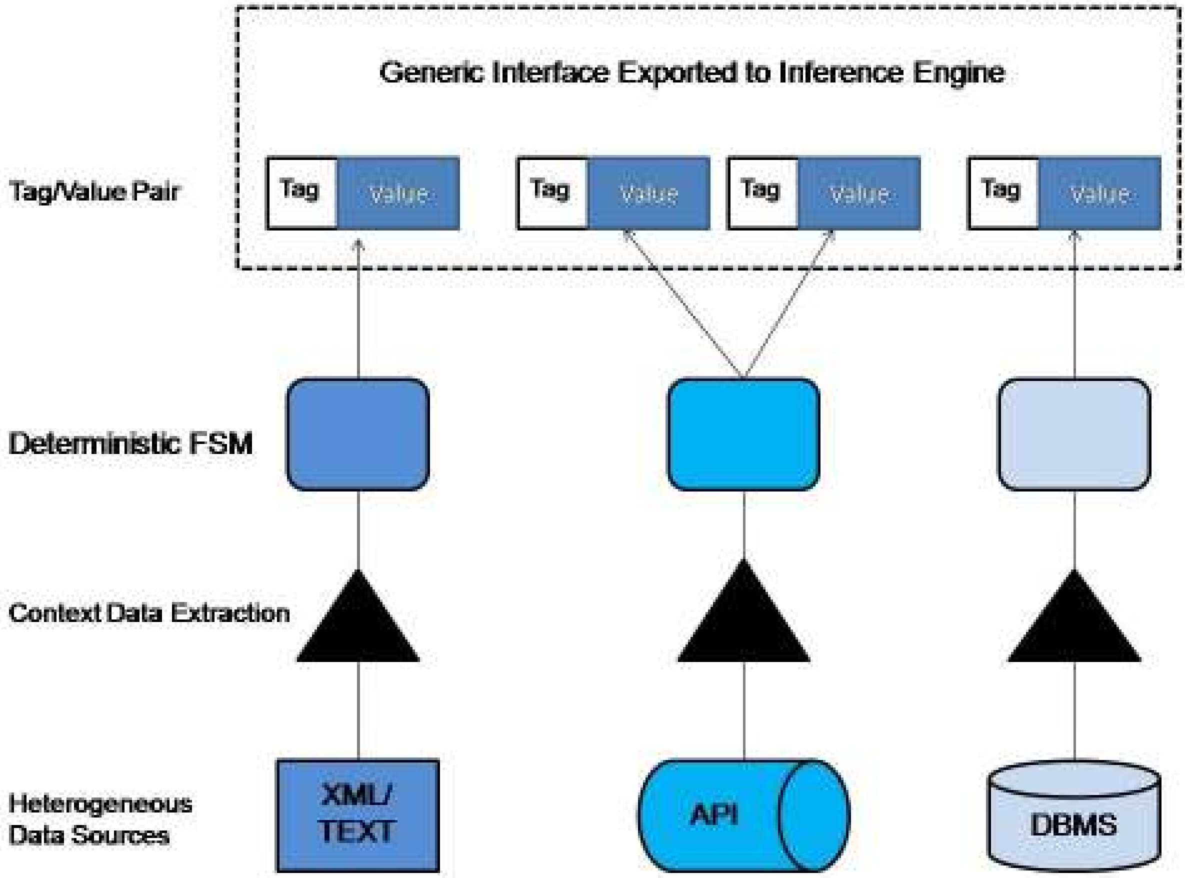


Figure 15: The Conceptual Context Modelling Framework

The diagram in the figure above shows the conceptual framework for modelling context information from the different data sources. The framework illustrates how contextual data can be extracted from structured as well as unstructured data sources. Dey suggested using context widgets [81]to uniformly extract context information. However, for this thesis, the different formats, protocols, hardware and data models are not important. The important factor is extracting context information from those data sources and developing a state machine that outputs a uniform representation of the context information.

The highest level with a “Tag/Value” pair represents the exported interface that can be used by the inference engine when it applies the rules. The information in this interface is explained below:

Tag = the descriptive name of the context variable, for example, NOISE or LIGHT.
Value= a 1-bit binary digit or Boolean, representing the state of the tagged context variable

As shown in the figure above, it is possible to derive more than one context variable from a context source. This decomposition has an important implication that irrelevant contexts can easily be *excluded*. Take the following states for example:

00 = Natural Light
01 = Fluorescent Light
10 = Dark

A typical state machine would then move among the three states, depending on the input from the light sensor. This is a crude example, but it can still be decomposed into two machines as follows:

1 = Natural Light
0 = Dark

AND

1 = Fluorescent Light
0 = Dark

When inputting the rules, the relevant “tags” are selected and actions defined for when the tag is true or false. In some instances it might not be necessary to consider whether the light source is natural or artificial, therefore one can consider “Natural Light/Dark” option only.

However, caution must always be exercised to ensure the variables are relevant to the task so that mobile resources are not wasted on irrelevant computations. The rules, as discussed in the succeeding sub-section, will determine the variables to use out of all the exported variables (in tag/value pair). Thus, neither the exported interface nor the inference engine needs any changes. This is made possible by the complete separation of the inference engine code from the actual rules. Before proceeding further, I will briefly formally define what a finite state automaton is.

Definition

In general, a state automaton is a model of the behaviour of the system [82]. A deterministic FSM is a 5-tuple [83] and the five components are explained below:

I. A finite set of states
As represented by the 2 states in my solution.

II. The alphabet
This is the values that can be taken in by the FSM.

III. A transition function
This function will determine which state is moved to next. In my simple solution, the state can only change to the other if input values meet the criteria.

IV. A start state
Both states can be a start state and this should not influence the behaviour of the machines.

V. A set of accept states
One of the 2 states will be an accepting state. The machines will wait in either state until there is an input that warrantees a change of state.

The reason for using a simplistic 2-state mechanism is to reduce the computational strain on mobile phones. Having every granule of context information in either of the two states simplifies the computational requirements, thus resulting in faster computations. A compounded context variable with more than two states will require at least a 2-bit binary representation and more than 3 rules as illustrated by an example 3-state FSM below:

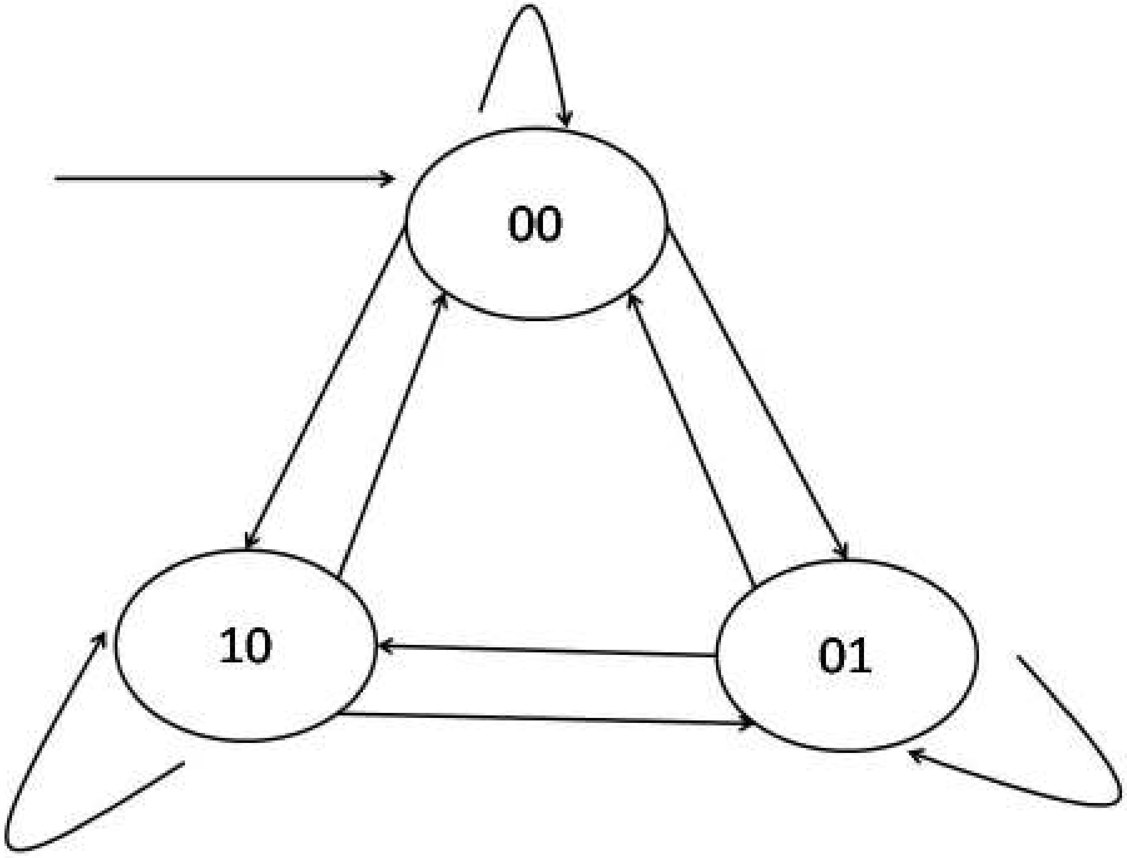


Figure 16: Skeleton of a 3-State Finite State Automaton

As shown in the figure above, the binary representation will need at least 2 bits. Consequently, the rules must also be at least 3 because if the state is not “01” for example, it can either be “00” or “10”:

```
IF NOT “01” THEN
        IF NOT “00” THEN        state = “10”
        ELSE                    state = “00”
ELSE    state = “01”
```

In contrast a 2-state machine-based rule will logically deduce that if it is not “1” then it is “0”:

```
IF NOT “1” THEN         state = “0”
ELSE                    state = “1”
```

Using this simplified approach I can illustrate how a state machine for the context variable noise could look like:

Example: *Ambient Noise*

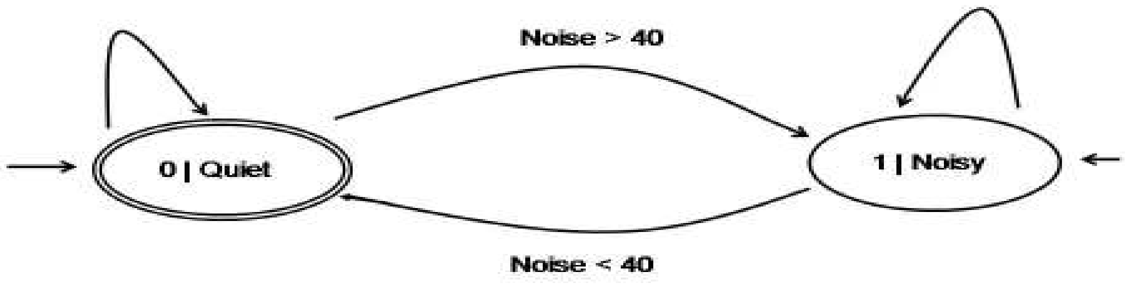


Figure 17: Ambient Noise FSM

In the figure above, the `Quiet` state is an accept state as illustrated by the double ring, although technically both are accept states. They both can be start states and a transition is deterministic once a state change is warranted. For example, if the start state were `Quiet`, the machine will keep waiting in that state until input determines a state change. An ambient noise volume, represented by an RMS value of greater than 40, will trigger a state change to `Noisy` as shown in the state transition table below. A timer is important for both states to ensure the machine is not stuck in one state forever [82]. The timeout period for each FSM will depend on the type of context information. For instance, changes between day and night are less rapid than changes in ambient noise conditions.

Table 7: State Transition Table

	< 40	**> 40**
Quiet	Quiet	Noisy
Noisy	Quiet	Noisy

Below is the table of the context information FSM:

Table 8: Context variables and associated states

Context Source	States (On/Off)		Possible Contexts
Ambient Noise	Noisy	Quiet	Football match, Conversation, Library, Lecture
Activity	Active	Idle	Walking, Sitting, Standing, Running, Exercising
Calendar	Busy	Free	Meeting in progress, Meeting in 5 minutes, Task
Phone Battery	Critical	Normal	Forgot to recharge in time
Time	Night	Day	Sleeping at night
Ambient Light	Dark	Normal	In a dark room (e.g. Cinema or bedroom), In the sun

The table above shows the context variables and their possible states. The FSM are meant to control the adaptation framework for handling the calls and messages. The adaptation framework will adjust the phone's settings in accordance with the environment. Analysis made by the middleware will result in context-sensitive output. Examples of such output are listed below:

I. Vibration

II. Volume Control

III. LED control

In addition, the context-sensitive output method can be in the form of specific actions taken according to specific contextual environments. For instance, the following are examples of context-sensitive action:

I. Diversion to voicemail

II. Deferring a message alert

III. Initiating a call

IV. Auto SMS to doctor or doctor's reception

5.4 Rule Design

This section introduces a way of constructing rules that are easily readable by both a human being and a machine. The rules should be expressed in an easily readable format because the rules might change or more rules may be factored in. This needs to have flexibility when managing the rules dictates that the rules be separate from the application code. The figure below illustrates how the context-sensitive framework separates the rules from the logic:

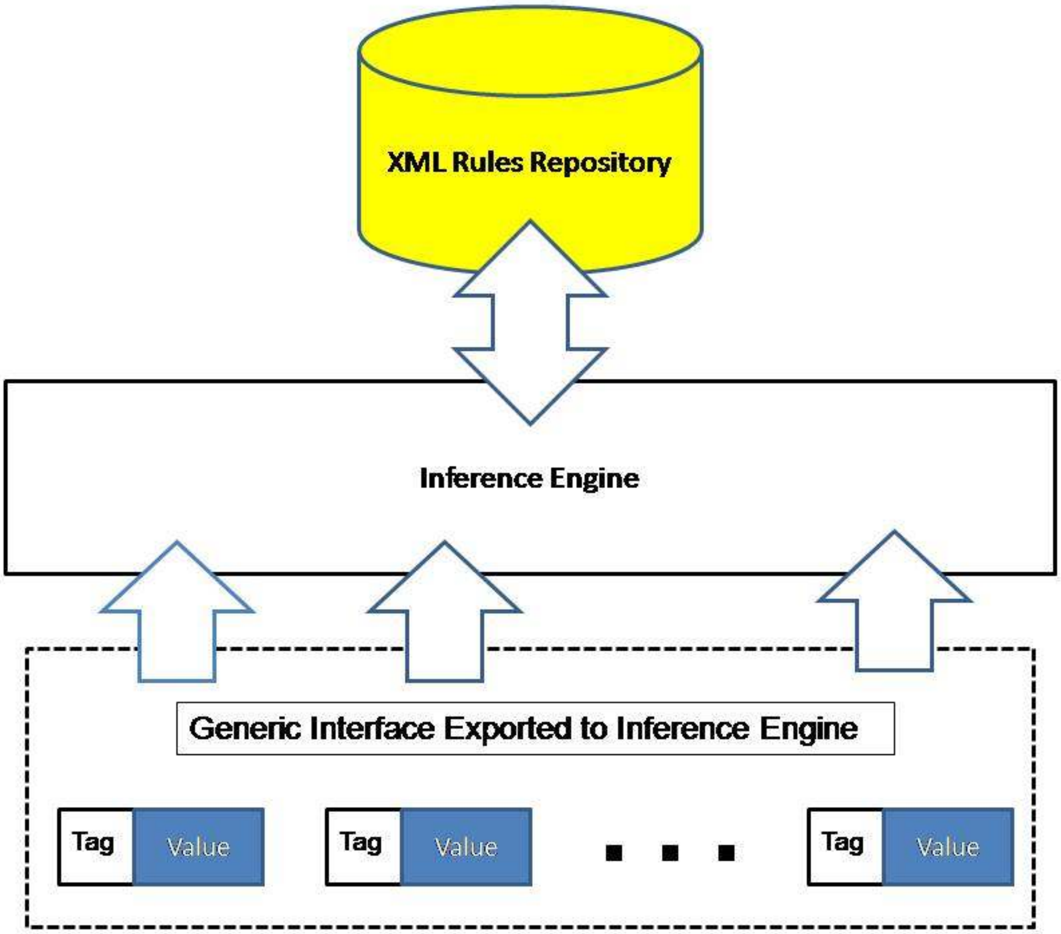


Figure 18: Separation of rules and application code

The figure above demonstrates the separation between the rules and the application code (Inference Engine). This separation creates the possibility of adding on more context variables without changing the application code itself. This is particularly important where rules can change frequently or where there is a real possibility of adding more sensor data. This is useful because different users will have different preferences and customizing or changing preferences should not necessarily affect the application.

To achieve this goal, an XML is used as the repository for the rules. XML offers a standard for storing the rules and this increases chances of interoperability and easy porting to different platforms. The actual rules are derived from analysing the available context variables and the relationships and priority among them as described in the next section.

5.4.1 Deriving the Rules

The rules are created using propositional logic to create deductive rules of inference. Using the 2-state paradigm I introduced earlier, I created a matrix for all the context variables and their possible states at any time. A full sheet is attached in Appendix-1A and Appendix-1B, but a sample shown in the table below illustrates an SMS/Popup event:

Table 9: Rule matrix for SMS/Popups

Noise	Activity	Calendar	Battery	Time	Light	Event: SMS/Popup
0	0	0	0	0	X	Normal beep alert
0	0	0	0	1	X	LED Flickers
0	0	0	1	0	X	Critical battery alert
0	0	0	1	1	X	Critical battery alert
0	0	1	0	0	X	Defer alert

Using the table above, the first step in expressing the rules will be to establish the premises from which I will derive the conclusion or inference. An example of propositions as taken from the above table is shown below:

Noise (N) = There is too much ambient noise

Activity (A) = The patient is hyper active

Calendar (C) = The patient is busy (in an appointment)

Battery (B) = The battery power is now critical (and not charging)

Time (T) = Time of day is night (not good for communication)

The first row in the table above is a rule that can be expressed in plain English as follows:
"If it is quiet, the patient is not engaged in rigorous activity, the patient is not occupied by a calendar appointment or task, the battery power is normal and it is day time then the arrival of an SMS/Popup can be alerted using the normal beep alert."

Obviously, it will be difficult to have the computer understand such high level language; the same row will be expressed using logic symbols as follows:

$$\neg N \quad \wedge \quad \neg A \quad \wedge \quad \neg C \quad \wedge \quad \neg B \quad \wedge \quad \neg T \quad \rightarrow \quad \text{Normal Beep Alert}$$

Where,

¬ = NOT logical operator and,

Λ = AND logical operator.

V = OR logical operator

→ = implies

Context-sensitive output methods are fewer than the possible situations; therefore, some conjunctures will result in the same output method. The table above shows two rows which have got the same output (critical battery alert) even when the context states are different. Instead of putting the two rules in the repository, a reduction mechanism is used to optimise the performance of the rule engine as discussed in the next section.

5.4.2 Reducing the Rules

In this section I introduce a shortcut to representing the rules. Since the context variables are represented by two-state machines, I will use ones and zeros using the propositions given in the preceding section. For example, the third and fourth row will be expressed as follows:

$0 \land 0 \land 0 \land 1 \land 0 \rightarrow$ Critical battery alert

$0 \land 0 \land 0 \land 1 \land 1 \rightarrow$ Critical battery alert

Having the above two rules resulting in the same output means we can apply an OR logical operator to the two rules and come up with one rule. Thus, the two rules can be expressed in the following composite rule:

$(0 \land 0 \land 0 \land 1 \land 0) \quad \vee \quad (0 \land 0 \land 0 \land 1 \land 1) \rightarrow$ Critical battery alert

The above step shows how we can combine the separate rules, but the following step will reduce the composite rule established above, resulting in a simplification of rule as shown below:

$$\begin{array}{lccccccccc} & 0 & \land & 0 & \land & 0 & \land & 1 & \land & 0 \\ & 0 & \land & 0 & \land & 0 & \land & 1 & \land & 1 \\ \hline \text{OR} & 0 & \land & 0 & \land & 0 & \land & 1 & \land & X \end{array}$$

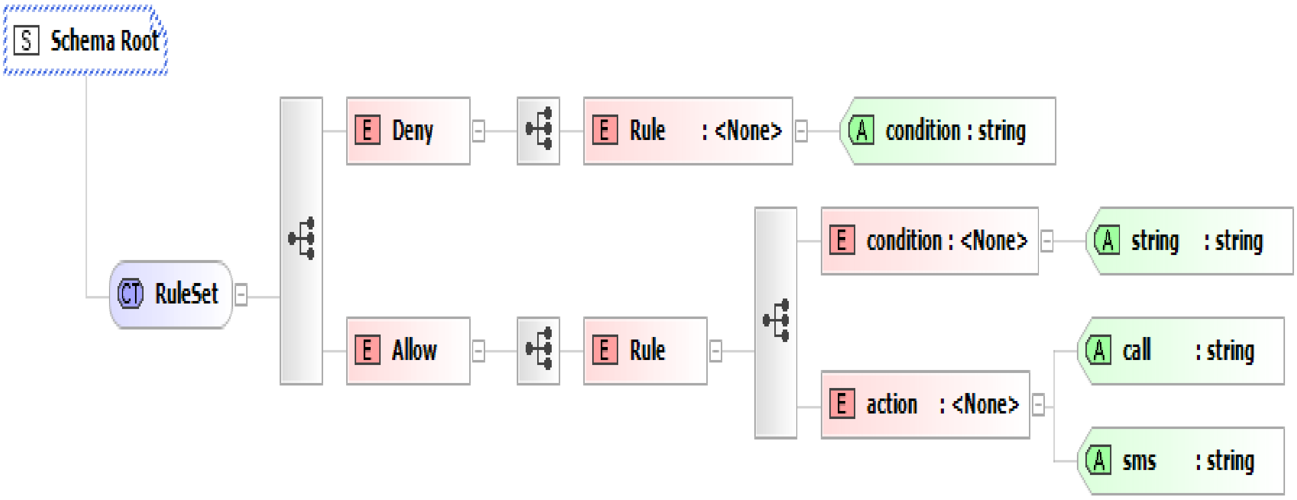
Applying the OR logical operator to context variables as done above usually results a context variable that I "do not care" about. In this instance the last variable, Time, is not significant and is overridden by the combination of all the other context variables. Instead of having the variable Time's state as 1, I do not care what state Time is in because it does not affect the outcome. In other instances, it is possible to have more than one variable insignificant. This happens whenever the context variable states do not match and yet the original separate rules result in the same output. Finally, the above analysis will result in this compact rule:

$\neg N \land \neg A \land \neg C \land B \rightarrow$ Critical battery alert

I use the letters as described in preceding sections to make it easy to conceptualise the new compact rule. Reducing rules to a compact form resolves any conflicts that may arise as a result to many rules fitting a criterion. Therefore, only one rule will fit a specific criterion.

5.4.3 Expressing the Rules in XML

As alluded to earlier, the rules are hard coded in an XML file. This semi-structured form is easily readable and can be adjusted easily to suit user preferences and to accommodate new context information. The schema definition of the XML rules repository is in Appendix-2A and the top diagram shown below shows the graphic layout, but the actual XML summarized layout is shown below it:



```
<?xml version="1.0" encoding="utf-8"?>
<XSDDesignerLayout>
    <RuleSet_XmlComplexType>
        <Deny_XmlElement>
            <Rule_XmlElement/>
        </Deny_XmlElement>
        <Allow_XmlElement>
            <Rule_XmlElement>
                <condition_XmlElement/>
                <action_XmlElement/>
            </Rule_XmlElement>
        </Allow_XmlElement>
    </RuleSet_XmlComplexType>
</XSDDesignerLayout>
```

Figure 19: Rules XML layout and schema definition

The rules are expressed in a binary string derived from the rules matrix discussed earlier. However, this time, only the final reduced rules are stored in the XML repository. As the schema above shows, each element will have two attributes, which are condition and action. The condition is the rule string and the action is the context-sensitive output channel prescribed.

The inference engine is responsible for processing the rules which implies finding the relevant rule fast. The rules are already reduced therefore the “seek” time should also be low. The inference engine is part of the main application code and will remain largely constant as the rules change and more context information sources are added. A conceptual overview of the context-sensitive framework is shown in the diagram below:

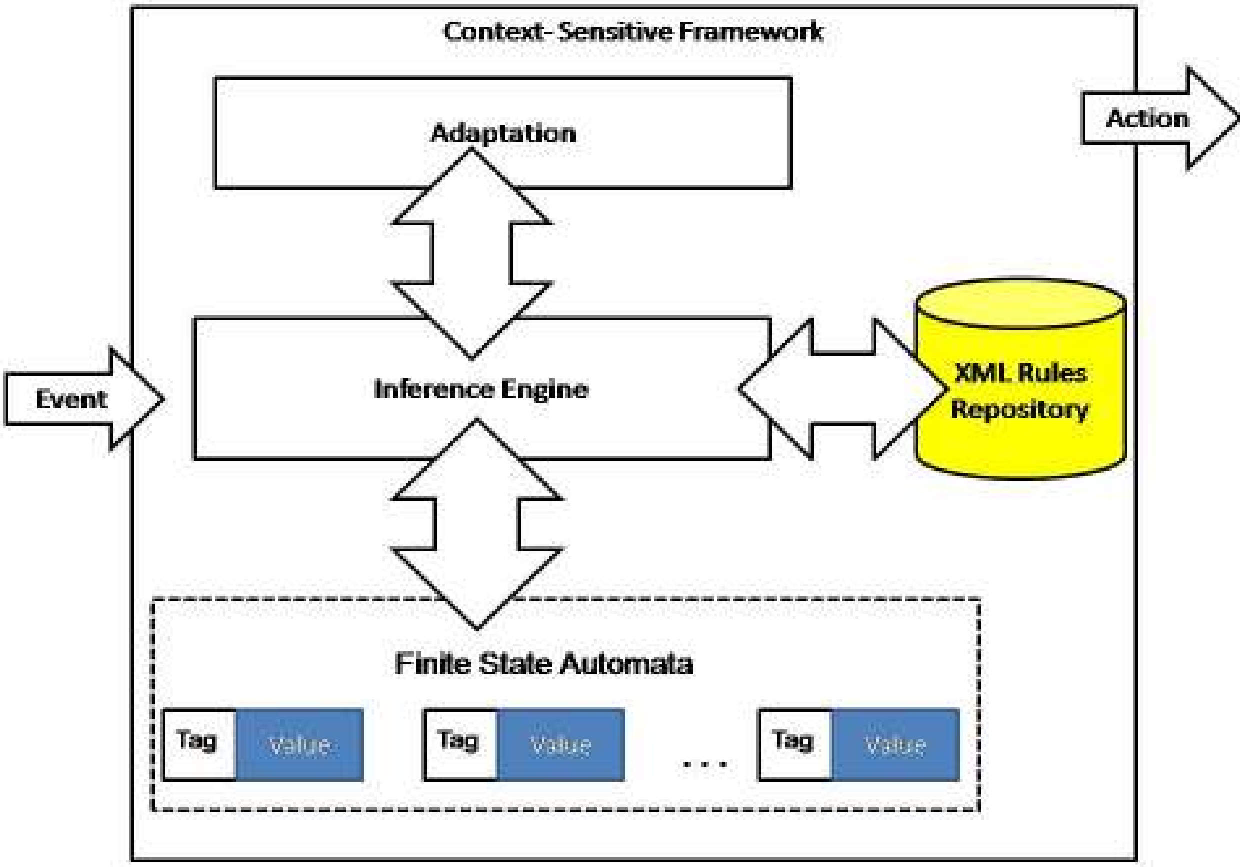


Figure 20: Context-Sensitive Framework Conceptual Overview

The figure above illustrates how the inference engine integrates with the rest of the system. The system uses forward chaining, that is, a data-driven approach. Forward chaining means the inference engine uses the live data or facts in main memory and searches the rules for the rule that best fits the read facts [84]. Input represents an event such as an SMS arriving. The arrival triggers the inference engine to apply the rules in the XML rules repository to current context states as provided by FSM data in main memory.

The results are used to adapt the mobile phone to the context and an output or action is generated. Adaptation means, for example, the device’s backlight will be adjusted to work with the current ambient light. The output can be figurative, for instance if the arrival alert is disabled then the alert is virtually deferred. This deferment is regarded as output. The rule processing details are discussed in the succeeding section.

5.5 Event-Condition-Action (ECA) Paradigm

I use the ECA paradigm which is based on an event-driven approach for constructing the inference engine. This is because the system depends on events to behave in certain ways. This approach aligns well with what Loke[85] describes as the three elements of a context-sensitive system:

A. Sensing (compared with monitoring for events in ECA paradigm)

B. Thinking (compared with checking conditions or rules in ECA paradigm)

C. Acting (compared with action in the ECA paradigm)

The ECA approach is illustrated by the figure below taken from [86]:

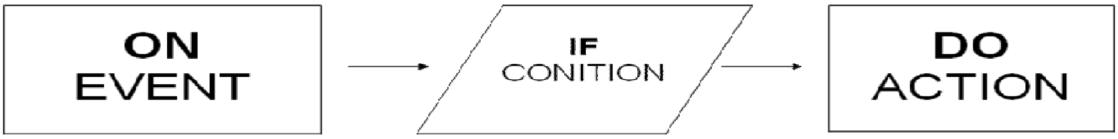


Figure 21: ECA model (Source [86])

The above figure illustrates how, when for example an SMS arrives, the systems checks the rules for handling the SMS in the current context and prescribes action as dictated by the rules. The main triggers (events) monitored are listed and described below:
I. Incoming calls

II. Incoming SMS

III. Popups

Popups include any other type of information that the patient must be alerted about. For example, a battery running low can be a trigger to generate an informative popup to that effect. However, there still remains a small portion that is not event driven and that works by polling. The polling mechanism is only used to adapt certain functions to the changing context. An example is the ambient light sensor which is initially triggered by a key press. The sensor state machine for ambient light will keeping polling for ambient light conditions and adjusting the display as long as the keypad is not idle.

5.5.1 Contrasting Alternatives: Event-driven vs. Polling

I do not use polling for other variables such as noise because adjusting the volume is needless without any incoming alert pending. The mobile phone is likely to be idle more often than it is in use and polling would only waste precious battery power. It seems more efficient to adapt to context when certain events occur rather than even when the device is not in use. To make the adaptation rapid, all the important data is kept in main memory for fast access.

5.6 Context-Sensitive Output and Action

This section describes the different output options available. The output options include adaptation and other context sensitive adjustments. The output can be adjusted just as the rules themselves, but the following table is a result of some experimentation and fairly represent the available context-sensitive output options on many kinds of mobile phones.

Table 10: Context-sensitive output

Output	Action
0	Ring
1	Beep
2	Vibration
3	LED flash
4	Voicemail
5	Alert deferred
6	Notify sender
7	Ring + Vibration
8	Ring + LED flash
9	Beep + Vibration
10	Beep + LED flash
11	Alert deferred + Notify sender

The table above summarizes the context-sensitive output channels. Each will be determined by the inference engine. For coherence with the rule set, a combination of any of the output channels is considered as a separate channel. In addition, a module will keep checking if there are any unread messages or missed calls.

When an alert is deferred, a module will keep checking for change of context until a suitable time is found. The sender of an SMS may be notified that the patient has not read the message despite having received it. It is also worth explaining why the patient may not be in a position to read the message at the time.

5.7 Modelling Integration with Self-Help Tool

Integration with the self-help tool is based on the premise that feedback is important to the patient. The feedback also includes educative materials sent by SMS to the patient. Another important aspect of the tool is the enhancements that increase the appeal of the application. Below I list the types of feedback information:
I. Graphical

II. Educational Tips

III. Motivational Tips

VI. Positive Reinforcement

Enhancements

The tool's enhancement possibilities are limitless and they include sound effects and graphical richness. Sound effects may be volume adjusted or muted altogether depending on the context. The educational tips are annotated with some sound, which can be played optionally. This is a useful feature that can be used when the patient is not in a meeting or other quiet place. It can also be useful for patients with visual problems because text on the small display can cause eye discomfort when reading.

By default the context of the patient will determine how the sound effects are used. For example, if the patient is in a low priority meeting, the sound effects may be muted entirely. The volume of the sound effects can also be adjusted according to the ambient noise. In addition, other contextual information can be considered. For example, when the battery is running low, it may be better to mute the sound effects in order to save on battery consumption.

In a similar way the tool's graphics can be adjusted as context changes. For example, colour and contrast can be toned down in dark places. While in low priority meetings, only summarized values will be shown but the full features can be enabled at the patient's most convenient times. A good example is the activity indicators given by the step counter, where information for the last 5 days is shown. A sample screen is shown below:

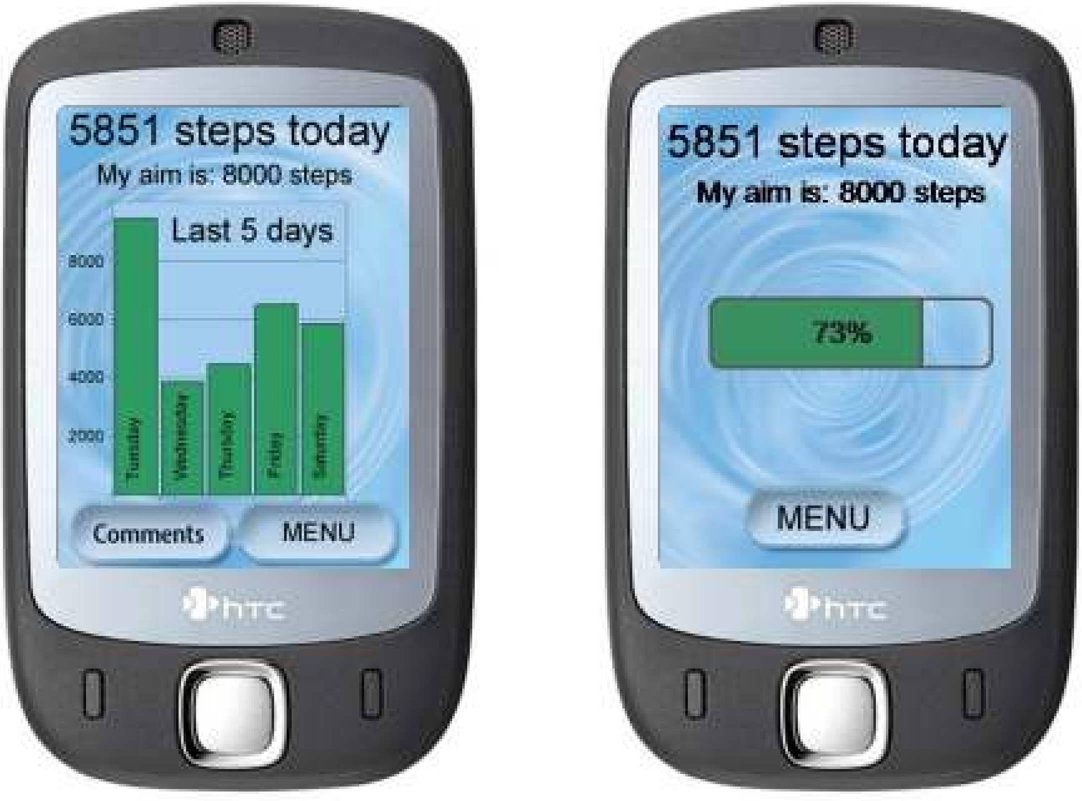


Figure 22: Physical activity screen (Detail VS. Summary)

The first screen in the figure above is taken from the diabetes self-help tool prototype [2] and shows the detail of physical activity over the last five days, while the second shows an example of how details can be pruned. When the patient is busy or in a meeting where just a quick snap will suffice, then the number of today's steps and the target may be enough. In most cases, the patient is likely to just take a peek at the graphical to see how far to the day's target. Therefore, the context will determine what the patient will see but with the option of switching views, of course. Thus, the system can make an intelligent guess at what the patient prefers, but the patient will always have the option of exploring further details.

Educational Themes

The educational themes that the tool uses may also depend on the patient's context. The time of day can be used to determine the kind of education and tips that the patient is likely to assimilate better at the time of day. For example, a tip about healthy foods can come at a time when the patient is planning the meal, thus helping the patient make wise decisions and meal choices.

Similarly, motivational tips for physical activity can be more effective during the day when the patient has time to act before going to sleep. The motivational tip does not have to come on a daily basis, but on whatever day it is determined, the tip must come in good time. The internals of whether the patient needs the motivation is determined by the self-help tool implementation. In some instances, the patient may be meeting all the set targets and deserving some positive reinforcement, the system can deliver such information around the time the patient takes measurements before retiring to bed.

Feedback
The regular feedback time for physical activity is optionally set to 10PM. Another regular feedback time is as and when the patient takes the blood glucose measurements. The patient chooses whenever to do this and there are no strict rules although it is generally recommended to take measurements at least once a day.

I can assume the patient is able to get some form of feedback after taking some measurements. The kind of feedback and graphical visualizations and enhancements will depend on the context. For example, the system will determine whether to give summaries or detail and whether to augment the graphics with sound effects or not.

5.8 Experiment and Testing Design

Experiments will be done by deploying the context-sensitive application on a mobile phone with the self-help tool. The controlled experiments involve simulating different contextual environments and noting how the mobile phone behaves. During the simulation, the accuracy of detecting contexts and the use of resources is recorded and the two key areas are discussed in the succeeding sub-sections.

5.8.1 Accurate Context Detection

The areas of interesting feedback are:

Before Deployment

1. Patient Information
1. 1 Age of the patient
1.2 Occupation
1.3 Knowledge level about diabetes
1.4 Years using mobile phone

2. Application
2.1 General feeling about context-sensitivity
2.2 Context priorities

After Deployment

1. Patient Experience
1.1 General feeling (likes/dislikes)
1.2 Accuracy of context detection
1.3 Satisfaction levels
1.4 Recommendations

2. Device Use
2.1 Memory usage
2.2 CPU usage
2.3 Battery usage

5.8.2 Resource Utilization

This refers to the effect of the framework on mobile resource utilization. This is important because mobile resources are limited and must be used efficiently. I will take measurements of the following variables:

- Battery life compared between those with and without the context-sensitive framework
- Average main memory usage of the framework

5.9 Summary

This chapter forms the core of my research and discusses the following main sections:

I. Context extraction
II. Context data representation and modelling
III. Rule design
IV. Inference engine design
V. Modelling integration with self-help tool
VI. Experiment design

The first section describes the chosen context sources and how the context information will be extracted from the respective sources. It describes the techniques for acquiring the raw data from the different sensors and information sources. Some information sources describe the status of the device and others describe the ambient conditions.

Context data representation and modelling is an important section that explains how and why context data should be discretized and appropriately grouped into classes that denote separable states. Then my individual direction is introduced where I prescribe the decomposition of context variables and use of an abstraction to model context information. The section then goes on to develop solutions based on established state machines problems rather than the individual and unique problems with each of the heterogeneous data sources. The advantages of this approach are:

A. Decomposed context variables enable exclusion of irrelevant or unnecessary states when constructing the rules

B. The 1-bit binary representation offers less time and space computation complexity, hence faster and resource-efficient computations.

Later in the chapter I describe the methods for processing the context information using the Event-Condition-Action (ECA) paradigm based on the event-driven architecture. I go on to describe the design of the rules and the method for reducing them for more efficient processing by the logic processing unit or the inference engine.

The last section of the chapter describes the integration part with the diabetes self-help tool. It prescribes certain enhancements that make the tool more easily adaptable to different contexts, such as sound effects and graphical detail. The chapter concludes by describing the design of the experiments.

CHAPTER 6

Implementation

The implementation is done using Microsoft's .NET and the Visual Studio Integrated Development Environment (IDE). This platform is widely supported my Microsoft and offers resources for fast and easy development. The use of very high level languages makes the environment easy to programme. It also offers support for lower level languages for implementing facilities not covered in the common libraries.

6.1 Programming Techniques

I mainly used C# language in the .NET environment and occasionally had to use Visual C++ for functions that I could not locate or were not supported in the .NET Compact Framework (CF). The main technique I used is the **`threading`** components in the **`system`** class.

6.1.1 Microsoft's .NET Environment

The prototype is developed using .NET CF 2.0 and the Windows Mobile 6 SDK. Using C# presented a slight limitation because, although the .NET CF is rich with libraries, many of the lower level controls are not implemented the way I wanted. However, .NET uses a powerful tool to access methods implemented in external libraries.

I therefore had to use the Platform Invoke (PInvoke)services [87] to access libraries with native C++ code. The process involves developing methods in C++ code and compiling them into a dynamic link library (.DLL). These .DLL files written in C++ are more portable and can be imported (PInvoked) easily and used within C# code.

Thus most of the components of the system were done entirely using C# code, but for lower level functionality I had to use PInvoke. A sample "signature" for using the PInvoke services is shown below:

```
[DllImport("context.dll")]
 private static extern void setBacklight(int brightness);
```

The `[DLLImport]` is provided by Microsoft's `System.Runtime.InteropServices` class and enables me to use exported methods in my "`context.dll`" library. The declaration indicates that the `setBacklight` method implementation is done outside the current namespace and in some other external library, hence the key term "`extern`".

In the prototype, the processing of rules, context information and partly context extraction are all implemented using C#. Extracting context information from more hardware based sensors and information sources requires native code such as C or C++. The advantage is that these languages are more portable and faster and more suitable for driving the hardware. Applications that need some driving of hardware are normally implemented using lower level code.

This tendency to shy away from managed code is because, while managed code does have many useful functionalities, it is not flexible and normally only caters for broad and popular requirements. Thus, developers with specific needs would normally have to develop their own native implementations. Below I summarize the different parts of the system and the approach I used for each:

Table 11: Managed Vs. Native Code APIs

Managed (.NET CF C#)	Native (.NET CF Visual C++)	API Reference
Calendar		POOM API
Battery		SystemState API
Clock		DateTime API
Microphone		WaveIn API
	Camera	DirectShow API
	Backlight	PowerManagement API

From the above table, battery and clock are hardware based components, but there are many APIs for accessing and controlling them so that I do not need to develop my own. For the camera function, seldom do users require to use a camera without the normal preview dialogue box like I wanted to. The environment.NET only provides limited camera functinality therefore I had to use native code to access the camera directly and get photos the way I wanted them.

6.1.2 Threading and Event Handling

In order to speed up the process of updating context machines, I use multithreading. Using threads improves performance even on a system with only one CPU. On single CPU systems there will not be any true parrelism but the efficient CPU time slicing will result in superior performance. An adavantage of using threads is that they use the same address space thus the spawned threads can all share data easily with the main thread. There is no threat of sharing violations because the spawned threads only share data with the main thread. The main thread only performs **`read`** operations and all **`writes`** are done by the respective threads that own the data.

I launch dirrerent threads for processing data from different context sources. This is because extracting and processing data from some sources will "block" the computation. Below I give the thread groups, with `Thread#1` as the main thread, and why threading was necessary:

Table 12: Thread groupings for context sources

Thread	Context Source	Blocking Possibilities
#2	Microphone	Takes at least 1 second on gathering the ambient noise samples
#3	Camera	Takes time to connect to and get data from the camera.
#4	Calendar	May fail to get read access to the calendar
	Onboard Clock	None worth considering
	Step Counter	May fail to get read access to the imported step counter data
	battery	None worth considering

As seen in the table above, I grouped the operations that are expected to be fast in one thread, and the slower ones in their own individual thread. Despite the difficulties in handling the errors in the main thread, this approach results in much faster overall processing. The main thread will keep watch of targeted events to signal before an update of context states is done. The diagram below illustrates how using threads can results in less computing time:

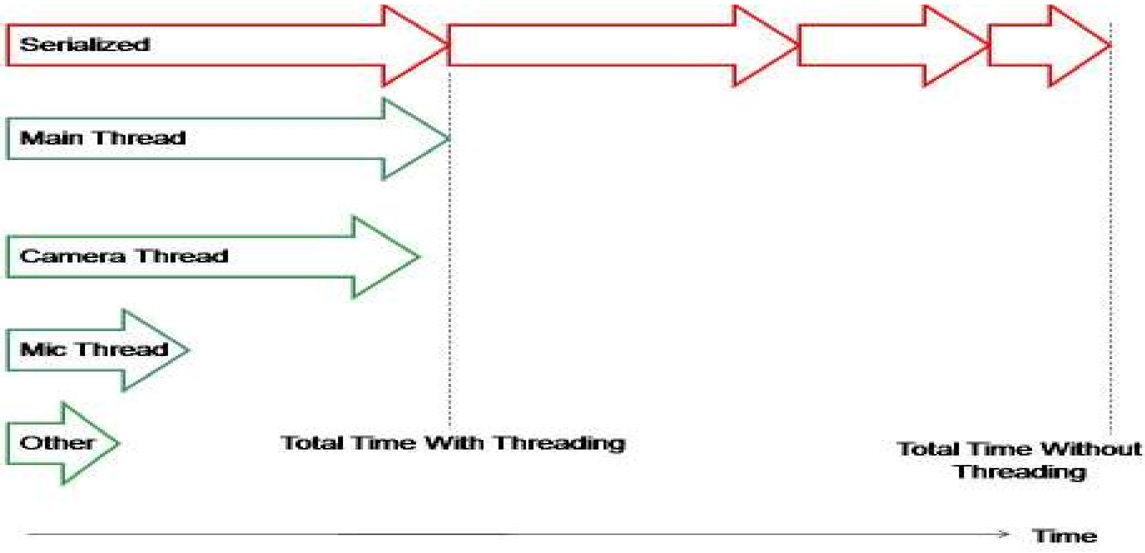


Figure 23: threading time line

In the above diagram, the red arrow represents the time if no threading is used. This means all the operations on the camera and microphone and other operations will be executed in sequence. The program acquires an image and perform calculations on the images and then when finished proceeds to acquire noise samples from the mic. This is highly inefficient as compared to using threads. By using threads I let the operating system allocate CPU time-slices to the different threads and this approach is more efficient because the different context sources do not block the whole computation. Thus, the total execution time will be significantly reduced.

The main thread will sit idle in memory until some events occur. Keeping the thread “asleep” conserves resources and the thread remains alseep until woken up. This means that the thread is virtually dead because it will not require any CPU cycles or time-slots and thus does not waste computing power. The follwing signature is used by the main thread to sleep while it is waiting to get signalled when some event occurs:

```
[DllImport("CoreDLL.dll")]
public static extern int WaitForSingleObject(IntPtr handle,
                                              int waitTimeInMsec);
```

The main thread waits on an event handle which is based on events that I created. When the handle on the event becomes true the main thread is signalled and then the threads for updating context states can be launched.

When the system must continually check for context changes, the framework uses a mecahnism that works as a “daemon”, or background process, without the need to wake up the device. The mechanism is PInvoked and has got the following signature:

```
[DllImport("coreDLL.dll",
EntryPoint      = "CeRunAppAtTime",
SetLastError    = true)]
private static extern bool CeRunAppAtTime(string pwszAppName,
                                                byte[] lpTime);
```

Using `CeRunAppAtTime` to shedule continuous scanning of context states presents an afficient and battery-conserving mechanism. The mechanism may require waking up certain devices on the mobile. For example, the speakers and camera on the HTC phone that I used for this project were switched off completely when in hibernation. Therefore, power requirements specified in the device driver would have to be increased before any sound could play or any pictures taken.

Continual checking is necessary when a message or call alert is put in the queue because the patient’s current context is inappropriate for interaction. The systems will need to keep checking the context until a suitable context is reached.

6.2 Framework overview

The framework presents an easy way to extract and process context information. The structure enables rules to be changed frequently without affecting the application code. This is especially handy where people have different preferences and rules can act only as an initial configuration of the behaviour of the system. The diagram below shows the generic data flow of the system:

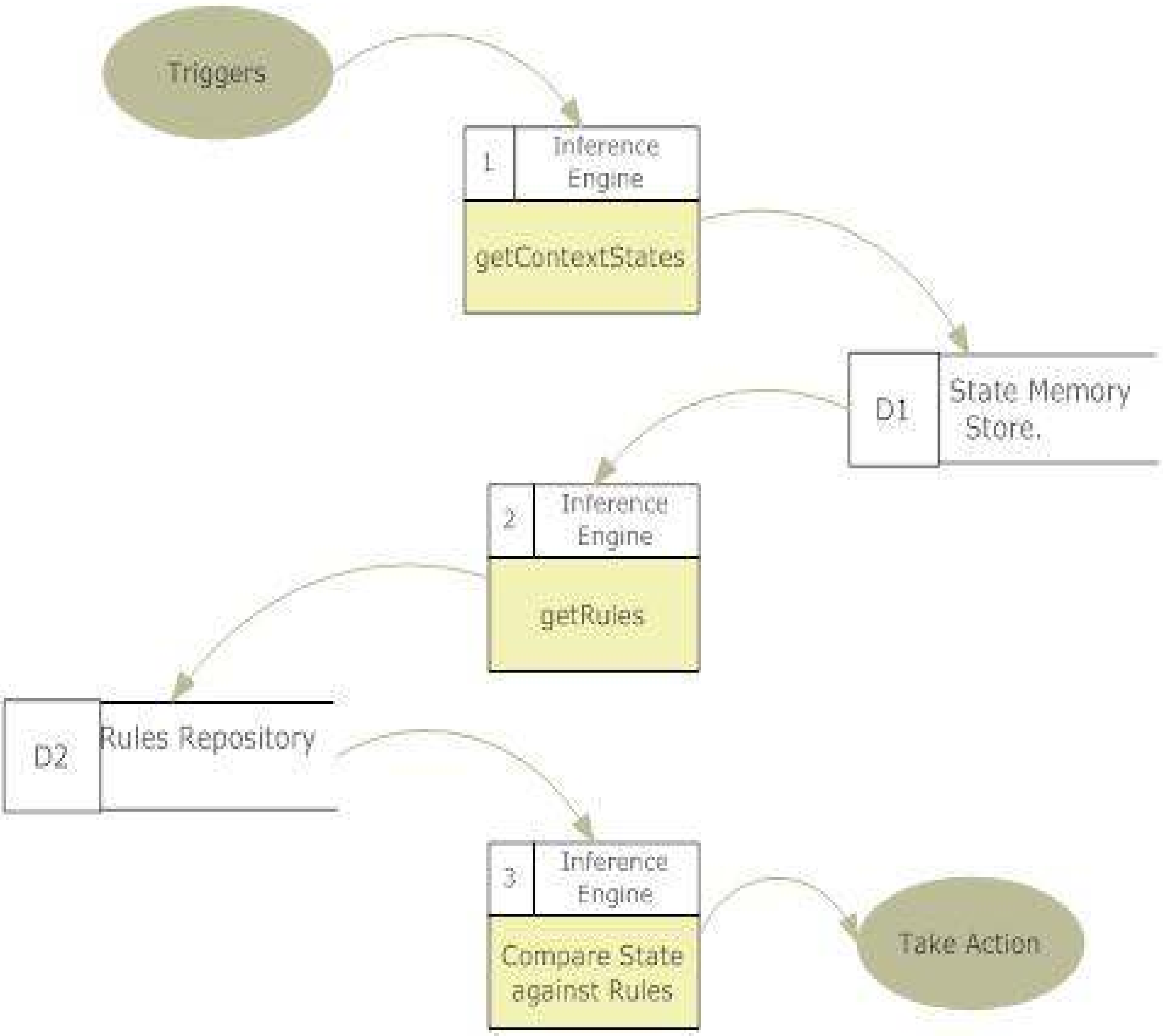


Figure 24: Dataflow Diagram

The diagram above uses SSAD convention[88] and shows the generic flow of data in the system. D1 and D2 represent the data stores on main memory that keeps the states and the rules, respectively. Triggers are the events that will occur such as incoming calls, messages and events generated by the diabetes self-help tool [2]. Comparison between the states and the rules will generate an output, represented by “take action” in the diagram above.

The figure above only shows the flow of data, but the actual system components are only implied. The main components of the system are illustrated by the diagram below:

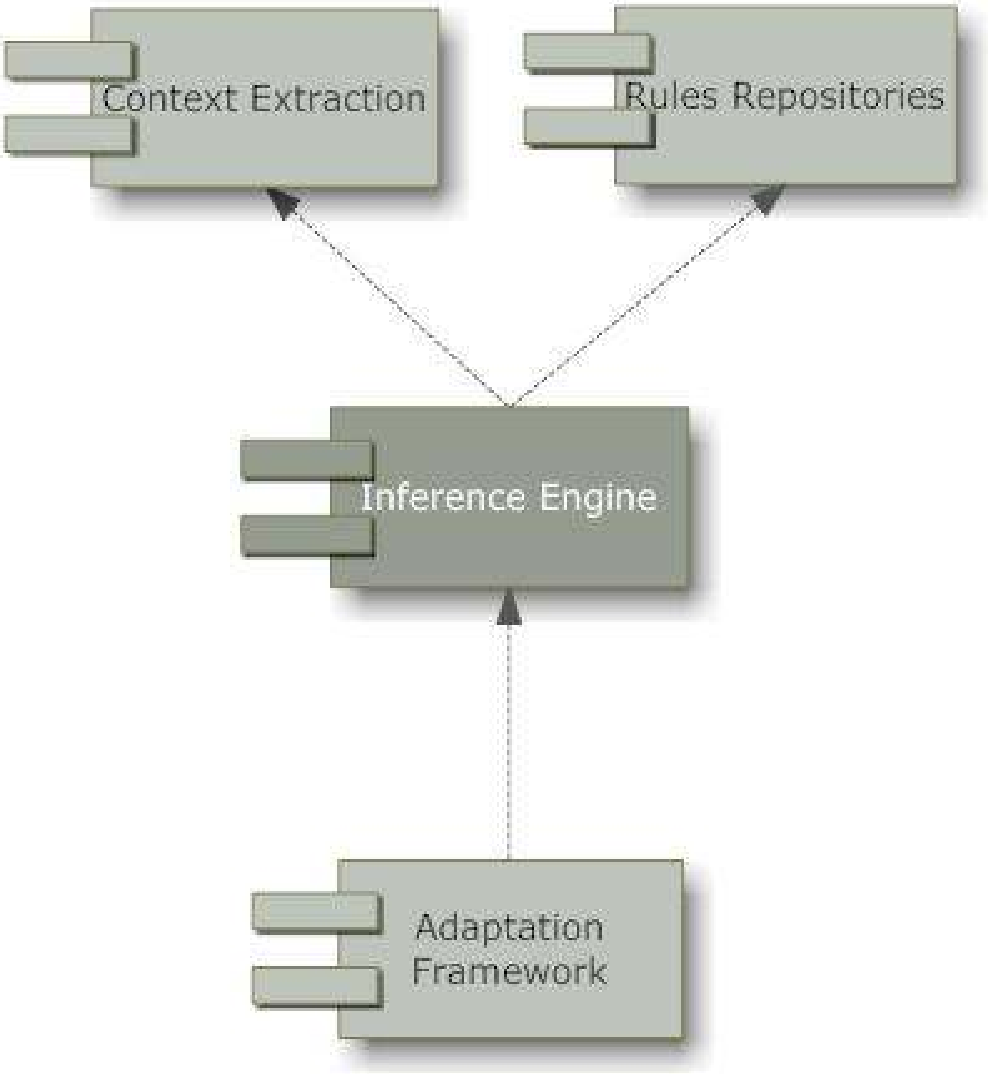


Figure 25: UML Components Diagram for the Context-Sensitive Framework

The figure above illustrates the four main components of the proposed context-sensitive framework. At the core is the system is the inference engine, which is the heart of the system. It gets input from context-extraction components and the repositories that hold the rules. The engine applies the rules to the context and generates a prescribed output or action for the particular situation. The adaptation framework depends on input from the inference engine for input in order to adapt the mobile phone to the changing situations.

The context extraction component implements various methods for getting context information from the identified sources. This is the creative part and involves maintaining a 2-state machine for every context component identified. Rules are implemented in XML in order to make them truly independent of the inference engine's code. In the remaining sections of this chapter I discuss the implementation details of these components.

6.3 Context Objects

This section deals with acquiring data directly from the targeted source. One of the objectives of this research is identifying potentially useful sources of context information housed within an ordinary mobile phone. When such a source is identified as potentially beneficial, then innovative ideas are applied to identifying possible contexts represented by the data. Below is a list of context information sources:

1. Calendar
2. Microphone
3. Camera
4. Clock
5. Battery
6. Step Counter

The sources are all available on ordinary mobile phones. A single context information source can be abstracted to represent several context scenarios. As discussed in the design chapter, a source of context information can therefore implement several state machines. The sample state machine implementation below determines whether a call can be taken at all given the surrounding noise or not:

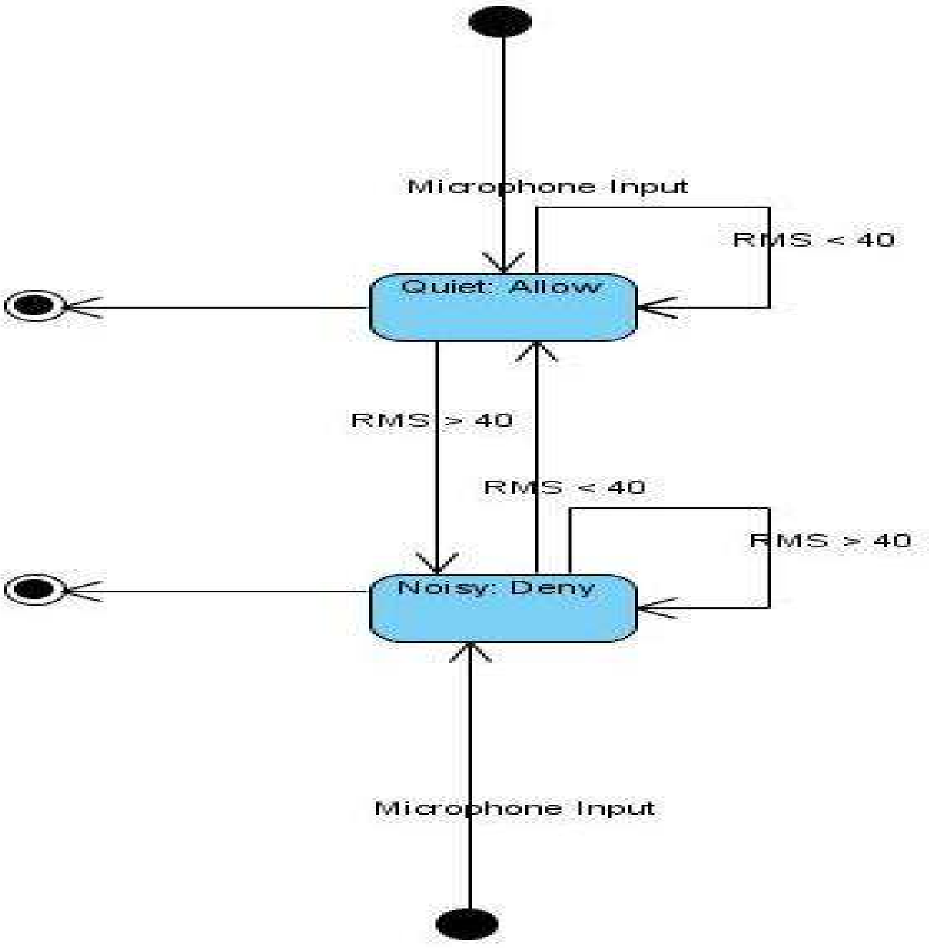


Figure 26: State Diagram for ambient noise

Using deterministic FSM with a 1-bit binary state representation is a painless process that simplifies the computation as well. In the prototype I stick to 2-state FSM in order to demonstrate the feasibility of the approach. A sample FSM algorithm is shown below:

```
Boolean noisy()
{
   bool noise = false;

    if (getAmbientNoise() > 40) // noisy if over 40
    {
        buff[NOISE_INDEX] = 1; //transition to noisy
        .  .  .
        noise = true;
    }
    else
    {
        buff[NOISE_INDEX] = 0; //transition to quiet
        . . .
    }
    return noise;
}
```

Snippet 1: FSM skeleton example for noise

The machine gets the ambient noise level as input and makes a transition depending on the level. A noise level that does not change above or below the 40 threshold will result in the machine not changing state. A transition is triggered if the level crosses the threshold.

The threshold is an arbitrary value that I placed, but it could be any number. The actual values would be prescribed by qualified personnel who deal with patient's hearing capabilities and how the frequencies can be applied to individuals. For the purpose of demonstration and proof of concept I just use the rule of thumb.

6.4 The XML Rules Repository and Inference Engine

The rules are implemented in XML and this is done in order to completely separate the rules from the application code. Changes in the rules will not affect the code in the inference (rules) engine. Firstly, the system processes “deny rules” before processing “allow rules”. This is only meant for efficiency because, when the context matches a deny rule, then no further rule processing is done. Samples of the rule structures are shown below:

```
<RuleSet>
    <!--First process rules for denying calls order of:
            General Priority.
            Noise Level.
            Calendar Priority.
            Time-->
    <Deny>
      <rule condition="0001"/>
      <rule condition="1111"/>
    </Deny>
    <!--Now listing rules for allowing calls order of:
           Noise.
           Activity.
           Calendar.
           Time-->
    <allow>
      <rule>
        <condition string ="XX1X"/>
        <action call="2" sms="2"/>
      </rule>
      <rule>
        <condition string ="X000"/>
        <action call="0" sms="X"/>
      </rule>
    </allow>
  </RuleSet>
```

Snippet 2: XML rule repository example

The rules within the `<Deny>` and `<allow>` elements can be altered and added as needed. However, care must be taken that the rules are complete, not conflicting and not redundant. A careful process of reducing the rules as described in the design chapter should be considered. Failure to do so will result in inconsistent rules and unpredicatable system behaviour.

The XML file is parsed by two methods in a seperate parsing class, one parsing the rules using XPATH string `(//Deny/*)` for deny rules and another using `(//Allow/*)` for allow rules. The two methods load the rules in memory as arrays of strings. Deny rules are packed in a one dimensional array while the allow rules are put in an indexed two dimensional array or hashtable as follows:

Rules	(0,i)	X000	X101	100X	111X
Call Action	(1,i)	2	5	7	3
SMS Action	(2,i)	2	X	6	4

The inference engine implementation is independent of the rules and processes the rules efficiently, using input from the state machines' data in a main memory buffer. The state machines generate a binary string representing the state or context at that particular time. This binary string is compared against the binary string in the rules. An example is shown below:

Context state string:

0	**1**	**0**	**1**

Sample rule strings:

X	0	X	X
0	X	X	1
0	1	X	0

The inference engine will fetch the context state string from state machines from memory and compare the first item to the first item in the first rule. In the example, the first item has a don't care value X and is ignored. Next, the second string is contrasted and they do not match, therefore the rule is abondoned for the next. The rule matches the context state string and the searh is immediately abandoned; the remaining rules are not processed.

The code uses an outer **while** loop instead of **for** loops to implement this. **For** loops could be used with a break inside, but using the while loop is far less complicated. Below is a pseudo code for processing allowing rules:

```
    while (!match)
           {
               Get the next rule
               mismatch = 0;
               for (int y = 0; y <= ruleSize; y++)
               {
                  if ("X") ignore item; //don't care value
                  if (mismatch) {mismatch++; break;}
               }
               if (mismatch == 0)
                   match = true;
           }
```

Snippet 3: Main "allow" rule processing pseudo code

When a match is found, then the corresponding output method or context-sensitive output is read from the XML rule file and returned to the calling method. Using this rule by rule search is faster than using a vertical search because a vertical search will yield many matches.

Deny rules are processed in a slightly different manner. The rules must be processed until a match is found or until all the rules have been processed. When a match is found, no further rule processing is done and the program exits the loop. This means the allow rules will also not be processed. Allow rules are only processed if there is no matching deny rule. The pseudo code for processing deny rules is given below:

```
for (int y = 0; y < denyRules.Length; y++)
 {
   Get the next rule
   mismatch = 0;
   for (int x = 0; x < 4; x++)
    {
      if (denyRules.Substring(x, 1) !=
                             FSM.denyBuffer[x].ToString())
       {
         mismatch++; //Current rule does not match
         break;      //so exit loop for current rule
        }
     }
    if (mismatch == 0)
     {
       match = true;
       break;
      }
  }
```

Snippet 4: Main "deny" rule processing pseudo code

6.5 Adaptation Framework

The framework is responsible for the actual adaptation of the mobile phone to the changes determined by the inference engine. The main adaptation areas and the corresponding adaptation processes are shown below:

Adjusting display backlight	Every 5 seconds while the keypad remains active
Adjusting volume	Only triggered once when an event occurs
Effecting changes in alert methods	Only triggered once when an event occurs
Other changes	As needed by Self-help Tool

Adaptation action is succeeded by some other action. For example, if notification is denied, the sender/caller can be notified why he/she cannot get through. Some of the information in the above table can also be illustrated with the diagram below:

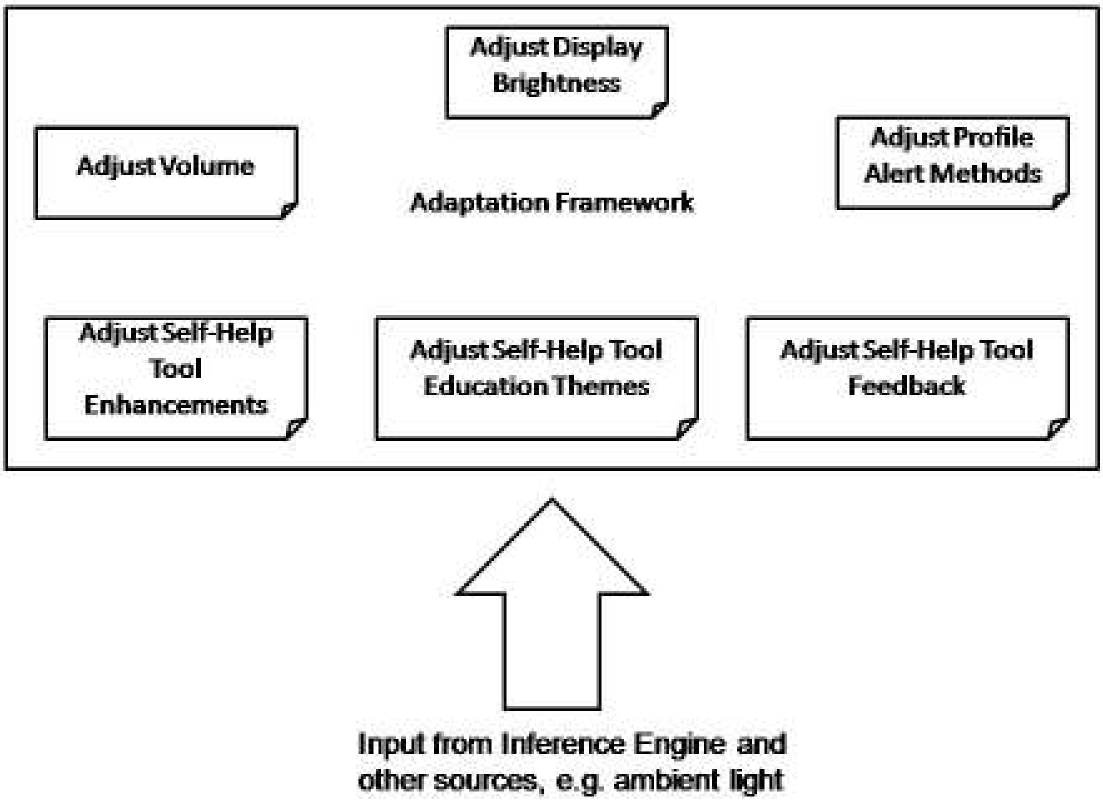

Adjust Display Brightness
Adjust Volume
Adaptation Framework
Adjust Profile Alert Methods
Adjust Self-Help Tool Enhancements
Adjust Self-Help Tool Education Themes
Adjust Self-Help Tool Feedback
Input from Inference Engine and other sources, e.g. ambient light

Figure 27: Adaptation framework

6.5.1 Main Strategy: Context-based Adaptation

Two sub-strategies are considered when adapting the mobile phone to the context changes. The aim of the strategies is to minimise battery usage and general resource usage. The first adaptation strategy is based on the event driven technique. This strategy implies that adaptation is only done when certain relevant events occur. This is a better approach that conserves energy than adapting the mobile phone to every change in context, some of which may not influence interaction with the patient.

The second strategy is only to adapt parts of the mobile terminal that influence the interaction between the patient and the mobile phone. For example, a patient may be in the sun and the screen is dim. When a call comes and the patient has auto-answer on, it may not necessarily always be wise to adjust the display brightness since the patient is holding the mobile phone to the head while in conversation. In such an instance, only the ring volume may need to be adapted to ambient noise. The patient may, however, want to check something on the phone after the conversion is ended and that's when the display would need to be adjusted.

This second strategy largely depends on the accuracy of context detection. The complexity of accurately detecting context makes it almost entirely impossible to implement strategy two. There are still limitations with using existing technology to accurately detect user contexts.

6.5.2 Adaptive Output Methods

I. Profile Notification Methods

Notification LEDs

The development platform allows us access to only one LED device for notification purposes, but in addition vibration is accessed as the second device in the notification LED (NLED) API. The functionality is accessed by PInvoking the NLED components in "`coredll.dll`". Other possible LEDs like the bright blue used by Bluetooth are hidden by the original equipment manufacturers (OEM).

This is not a limitation because the blue light is very bright and might easily disturb the patient anyway. The dimmer amber LED flash is especially useful in the night when the phone volume is off. The LED will flash to indicate unread messages, both SMS and voicemails as normal. In addition however, the LED will flash when the self-help tool has some patient interaction needs.

Sounds

The actual sound files are left unaltered so that the sounds remain what the patient originally set them to be. Only the volume is adjusted, the beep and ring tone remains unaltered.

Vibration

Vibration is implemented as part of .NET in the NLED API and I therefore use the functionality from managed code. The following code shows the code for activating vibration of the motor within the phone:

```
NLedSetDevice(nsi.LedNum, ref nsi);
```

The first argument is the device number and normally equals “1” on devices that implement vibration in NLED. A normal notification LED device number, as discussed in the preceding section, would usually be “0”.

II. Volume Adjustments

Adjusting the volume is only a matter of changing the volume registry key which is read by the speaker’s drivers. The key on most Windows mobile phones is a DWORD named “InitVol” in the following path:

```
[HKCU]\Controlpanel\SoundCategories\Ring
```

However, a problem arises when the volume must be adjusted after certain events occur. For example, when a call comes in, the phone system will take precedence and the speaker driver will not get a chance to refresh the volume levels.

A work around I used was to mute the normal sound and play another sound file. The settings for the other sound file will not be affected by the phone system. However a glitch that must me over come is increasing volume of the second sound file. On most mobile phones the phone system will normally mute all other sound files, but .NET now allows us to attenuate or even leave the second file completely unaffected. This can be done by tweaking the phone system to leave sound playing activities unaffected, using the `gainclass` implementation [89].

III. Display Brightness

I encountered some problems in trying to adjust the display brightness. Apparently, the drivers are developed by the OEM and APIs for accessing the display drivers are not always open. The basic procedure is to change the two brightness registry keys; ACBrightness and Brightness. The former is for brightness when the phone is AC powered while the latter represents DC or battery power. These keys are of type DWORD located in the following path in the registry:

```
[HKCU]\Controlpanel\Backlight
```

I created a handle on the “`BacklightChangeEvent`” which is fired when the keys are adjusted and asks the display driver to update and refresh the display. This general procedure would not normally work where the OEM has hidden the implementation details of the display driver. It is also worth mentioning at this point that the event is only fired when the device “wakes” up. I use the `WaitForSingleObject` described in the threading and event handling section and using the power saving machenism discussed earlier as follows:

```
//Declare the constants for handling the wakeup event

private const int NOTIFICATION_EVENT_WAKEUP = 11;
const string _eventName = @"ON_WAKEUP_EVENT";
const string _eventNameString =
             @"\\.\Notifications\NamedEvents\" + _eventName;

CeRunAppAtEvent(_eventNameString, NOTIFICATION_EVENT_WAKEUP);

/*In a separate thread create a handle on the event and wait
for the device to wake up */

IntPtr hEvent = CreateEvent ( IntPtr.Zero,false,false,
                                   "ON_WAKEUP_EVENT");
WaitForSingleObject(hEvent, INFINITE);
```

Snippet 5: OnWakeUp event handing sample

IV. Self-Help Tool Adjustments

Adjustments of the behaviour of the diabetes self-help tool are central to the whole process. The diagram below shows the summary of how the framework can enhance the self-help tool:

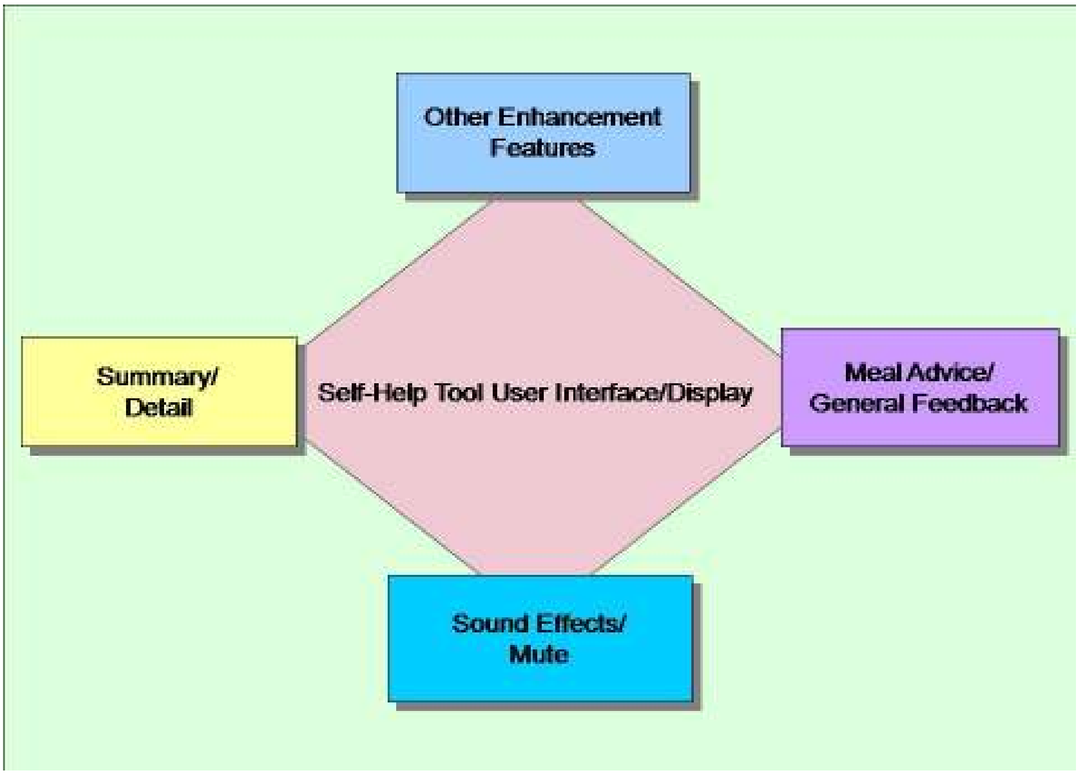

Other Enhancement Features
Summary/ Detail
Self-Help Tool User Interface/Display
Meal Advice/ General Feedback
Sound Effects/ Mute

Figure 28: Self-Help Tool Enhancements

The context-sensitive framework provides an interface that export context information as services, thereby providing contextual information to the self-help tool. There are basically two interaction scenarios between the self-help tool and the general context-sensitive framework:
I. Interaction Requests
II. Graphical User Interface (GUI) or the actual display

Interaction requests are made out by the tool and are handled directly by the context-sensitive framework, while the appropriate display detail can be determined by checking the share context information.

Therefore the design of self help tools becomes much simpler because developers will not care about the internal workings of the context-sensitive framework. The developers can concentrate on the core issue of disease management functionality. This means the developer should not care about whether sending messages at any time is suitable. The developer will just send the messages as per generic clinical requirements. The context framework will be responsible for the right timing and right notification methods.

6.6 Experiments & Tests

The first major goal of the experiments is to determine whether the framework can, to some acceptable magnitude of accuracy, determine the patient’s context. The behaviour of the self-help tool will depend on the detected context. The second major goal is to determine whether the framework conserves battery power, CPU and memory resources on the mobile phone. In the succeeding paragraphs I describe how I conducted the experiments.

Prerequisites
- An electronic calendar must be used to schedule events, appointments and meetings
- Windows-based mobiles will be used because the program may not be easily portable to other platforms.

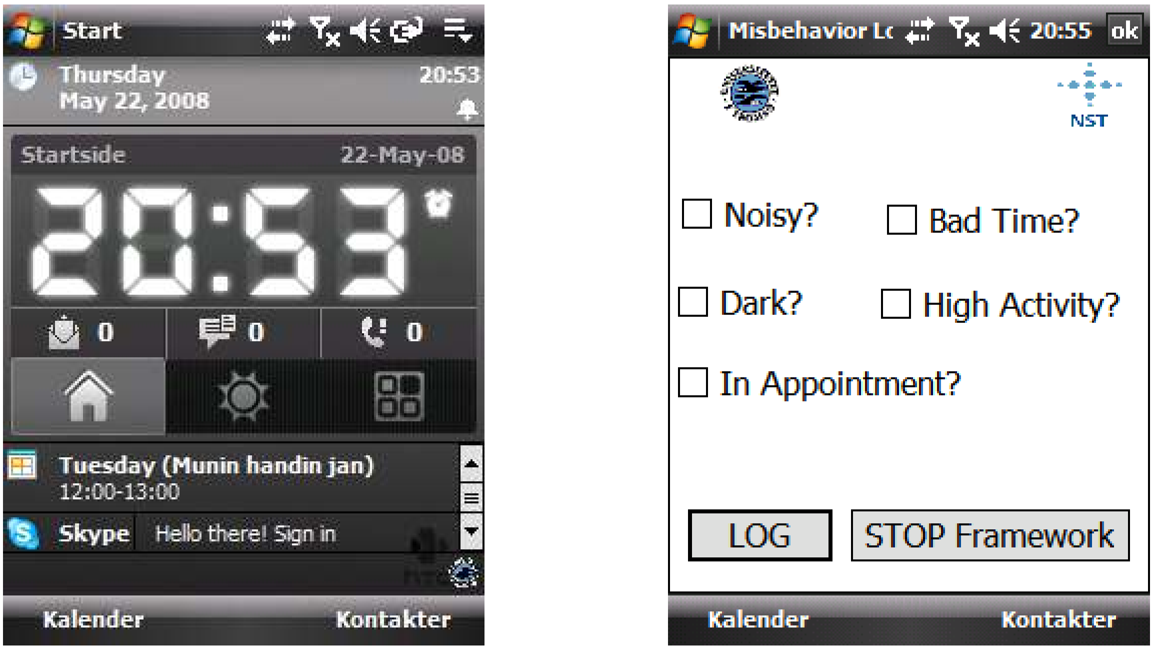
Sample
Norwegian law forbids experimenting with patients without prior permission and getting the permission would be time consuming. Instead I used two colleagues as the sample in order to get some practical feedback. In addition, I perform controlled experiments, subjecting the mobile phone to different combinations of context variable states.

Procedure
The contexts used were based on the variables that I earlier on deemed as important as follows:

- Varying light conditions
- Varying activity
- Varying the time of day
- Varying noise conditions
- Varying calendar tasks, appointments and their priorities
- Varying the battery levels

Because I used an iterative approach and separated the rules from the processing logic, I could easily modify, add or remove some of the rules to suit a situation that was overlooked during design. I could also change rules to modify the phone's behaviour in certain contexts.

I also tested the framework on two of my colleagues to confirm that the context-sensitive framework was operating correctly. The selected colleagues used the system for a 5-day week. They recorded calls that were alerted to them inappropriately, that is, in a way they did not like or expect. For instance, if the person was in a library and the phone rang loudly, disturbing the person and the people around, the user would use the screen below to log the reason:



Start
Thursday
May 22, 2008
20:53
Startside 22-May-08
20:53
0 0 0
Tuesday (Munin handin jan)
12:00-13:00
Skype Hello there! Sign in
Kalender Kontakter

Misbehavior Lc 20:55 ok
NST
□ Noisy? □ Bad Time?
□ Dark? □ High Activity?
□ In Appointment?
LOG STOP Framework
Kalender Kontakter

Figure 29: Feedback mobile phone screenshots

The logging program remained running in the background and displayed a tray icon as shown in the picture on the left (the bottom right) in the figure above. The picture on the right in the figure above shows the feedback form. The user would only have to press "LOG" in order to log the reason.

All call times are logged but for privacy reasons I do not take any further information about the calls. The context states at that particular moment were logged and the, apparently inappropriate, method used to alert the user were also logged.

Resource Consumption

The experiments are carried out on two phones, one running the framework and another without the framework as control. Both phones have the same configuration and programs running at any time. The only difference is the framework running on one of them.
The following aspects were measured and recorded:

- Main memory consumption is measured when the framework is running
- Battery consumption is measured against the control phone over a period of time
- CPU time is measured by a small program that uses time-stamps on both phones.

6.7 Summary

The chapter starts by detailing the specifics of the programming environment, that is, using .NET CF and the Visual Studio IDE. The chapter describes multithreading as the main technique used to optimise the performance of the frame work. The threading techniques and how they were helpful is fully discussed at the beginning.

The chapter then illustrates the overview of the system using a dataflow and a component diagram, before giving the details of modelling context information. The chapter also describes the implementation of the rules in XML and provides the algorithm for the inference engine logic in sampled elegant pseudo code.

Further down, the chapter describes how the output of the inference engine logic will be used to adapt the applications on the mobile phone, with specific details on the self-help tool. It describes how the context-sensitive framework exports context information as services which are used by the self-help tool to adjust the enhancements and feedback mechanisms to suit changing context.

The chapter concludes with describing the controlled experiments and also the feedback got from the 2 people who used the framework to verify the controlled experiment results.

CHAPTER 7

Discussion

This chapter deals with all the major findings and the possible interpretations of such findings. Initially, discussions are based on the survey that was conducted and later, the controlled experiments that were carried out. The results of this research are based on two main criteria. The first criterion is the correct and accurate detection of context while the second concern resource utilization on resource-limited mobile phones. Accurate detection of context reflects on how well the context-sensitive framework was constructed, while efficient use of resources reflects on how feasible the framework will be.

7.1 User Preferences Survey

To have an idea of what the potential users feel about the different contexts I conducted a survey on real diabetes patients. The survey tried to gauge at the different reactions in different situations and to see whether potential users would be happy to have a "smart" phone or whether they were impartial about the idea. The results of the survey are summarized in the succeeding section.

A copy of the sample survey form is attached in Appendix-4A and the results flyer is in Appendix-4C. The survey, conducted on 11 people with diabetes, yielded the following results summarized in the figure below:

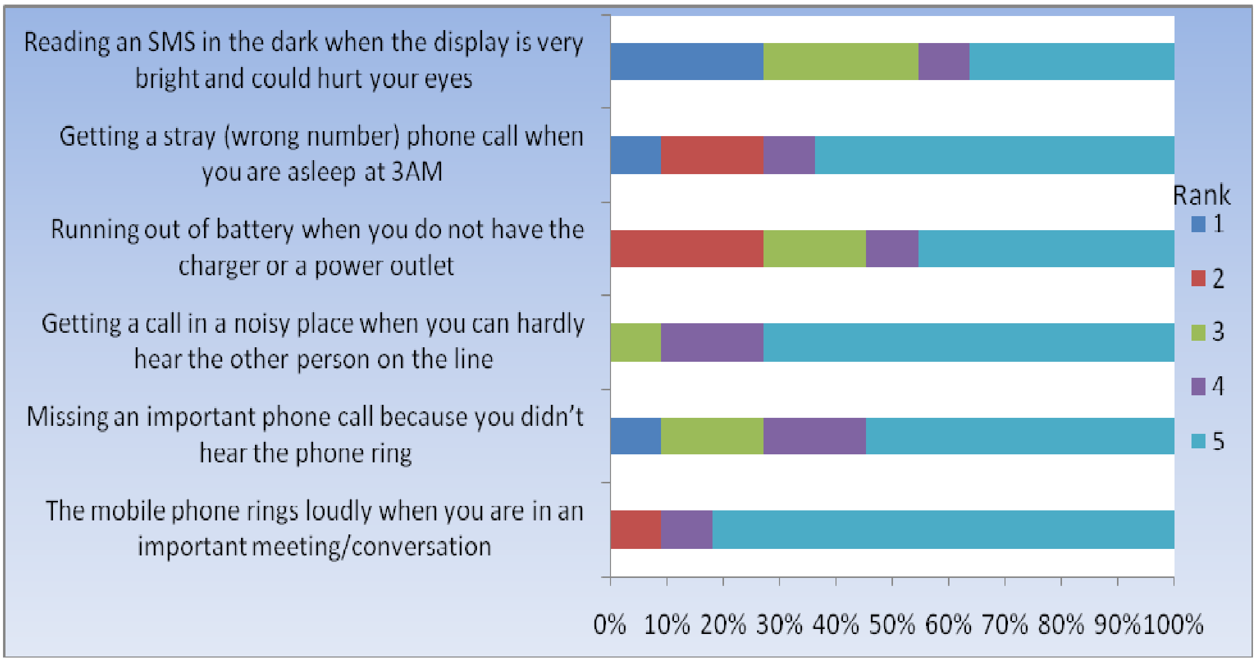


Figure 30: Users' perspective on contexts (Rank 1 = most unpleasant situation)

The questions in the figure above are related to the different context variables and unpleasant situations. The respondents were supposed to appoint a rating to each situation to reflect the most unpleasant situation. This would also be a reflection on what the patients found most important to them. Using their ratings and the simple formula below a priority list can be constructed as follows:

Σ `(Rank * Number of people)`

1. Display backlight
2. Battery power
3. Time of day
4. Ambient noise
5. Calendar

Summary
Generally the respondents were keen on having context-sensitive functionality, and their participation in the survey helped me focus on issues that are actually important to them. It is interesting to note how the sample cared very little for the phone to ring loudly when they are in an important meeting. It may be because the age group is retired or near retirement and no such thing as an “important meeting” exists anymore. Rather they are more annoyed by trying to read an SMS when the backlight contrast is wrong. This can be attributed to the fact that eyes may become more sensitive as people age. Naturally, the sample finds adjusting the backlight the most useful functionality. Therefore, such functionality would have to be taken seriously since we expect the patients to use the phone more frequently.

Age group
Most of the patients in the sample group are over 55 years old. This implies that most of the people targeted for the self-help tool are people past their middle age.

Experience using electronic applications
Of the 11 sampled diabetes patients 3, representing 27%, had used some form of electronic disease management application. This number is indicative of the fact that electronic disease management applications are yet to be used on a wide scale. It is also interesting to note that two of the three were over 65 years old. They did not use the applications for long either, clearly confirming the fact that patients tend to abandon use of disease managements application within short periods.

Experience using mobile phone
It goes without saying that all the respondents had used a mobile phone for more than two years continuously. This implies that the mobile phone has become an accepted gadget in people’s lives and has the potential to become a useful healthcare gadget as well.

Using appointments on mobile phone

Only 4 out of the 11 (representing 36%) people in the sample said they used an electronic calendar, either on a desktop or mobile phone. The question regarding the use of an electronic calendar was important to see whether scheduling appointments and tasks was important to them and, apparently, it was not.

Exercise routines

Most of the respondents normally walk or jog when they do their physical exercise. These forms of exercise are not vigorous and are easy to detect using the step counter.

7.2 Preference Configuration Template

The possibility of separating the rules from the application presented a good opportunity to customize the rules according to any preferences. The phone will only behave in a way that corresponds to and configured in the rules. This implies that the core of the rules engine (inference engine) was well constructed and together with the event-driven mechanisms, the context-sensitive framework exhibited excellent results. Below I tabled the different context variables and their practical levels as reflected in the experiments:

Table 13: Results: sample configuration template

Context Variable	Calculated/Obtained Values	Set Values	FSM State	Rule
Noise - Volume	0 - < 1	1	0	Allow
	1 - < 5	2	0	Allow
	5 - < 20	3	0	Allow
	20 - < 40	4	1	Allow
	> 40	5	1	**Deny**
Light - Backlight	0 - < 10	1	1	Allow
	10 - < 50	2	0	Allow
	50 - < 150	3	0	Allow
	> 150	4	0	Allow
Calendar	In Progress - Normal Priority	-	1	Allow
	In Progress - High Priority	-	1	**Deny**
	Not In Progress	-	0	Allow
Time	00:00 - 05:59	-	1	**Deny**
	06:00 - 23:59	-	0	Allow
Battery	< 10	-	1	Allow
	> 10	-	0	Allow
Activity	Walking	-	1	Allow
	Jogging	-	1	**Deny**
	Stationary* (sitting/standing)	-	0	Allow

*Also includes few insignificant steps, i.e. less than 6 steps.

The summary of the configuration given in the template above was determined to be optimal by trial and error. Thus the assertions are highly subjective and the actual configurations will depend on user preferences and the individual's cognitive sensitivity. The table is explained in the succeeding paragraphs.

Context Variables

The context variables are the context information used and were identified as important earlier in this research.

Calculated/Obtained Values

These values are either calculated or obtained using specified APIs for reading the information from the context sources. For ambient variables (noise and light), the information was calculated from sensor input. For instance, the noise was calculated as the Root Mean Square (RMS) of PCM data in the `.wav` sound samples collected from the phone's microphone. In contrast the appointments and tasks are read directly from the calendar using the Pocket Outlook Object Model (POOM) API.

Set Values

These values are set on the phone according to the calculated values and are dependent on the type of phone. For the HTC Touch phone that was used, the levels for noise are from 1 to 5 and the levels for the backlight are from 1 to 4. The other variables only include reading existing data from the sources.

FSM State

This is the 1-bit binary digit that denotes the state machine's state at the corresponding calculated/obtained values. The value 1 is a "high" state and refers to the lesser desirable state for interaction. For instance it refers to high noise, very dark or high activity, as the case may be.

Rule

This column refers to the type of rule that will be applied to the attribute. They determine the deny conditions such as when there is a high priority task or appointment in progress, or when the noise index, RMS, is over 40.

User Feedback

The 2 people who used the system gave me positive feedback that I used to evolve the system. One of the people was a colleague involved with the wider research group while the other was a friend who has no knowledge or input in my research. The friend thought the system was useful and wondered why the system "*does not come on every mobile",* but then again, that is what friends are supposed to say. The colleague gave me useful insight into problems, but she said the prototype could detect useful contexts accurately.

Such feedback affirms the controlled experiments and demonstrates how the framework can increase user satisfaction with mobile applications in general.

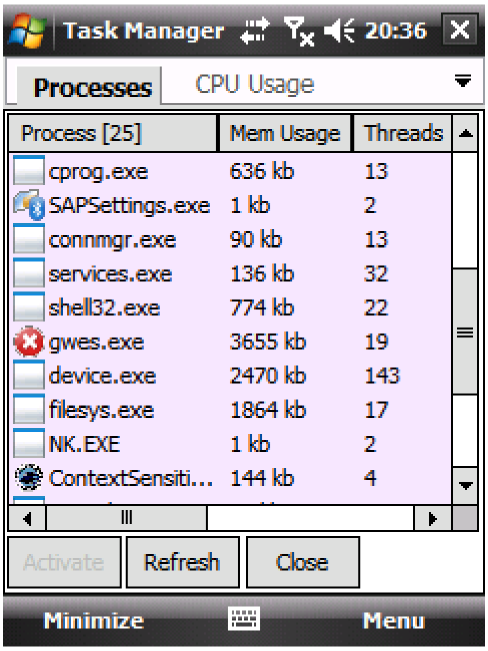
7.3 Resource Utilization

This section discusses the results of the experiments done on the mobile phones to determine the effect of the framework on the mobile phone's resources. The results of the key performance areas are discussed in this section.

Main Memory Consumption

The consumption of secondary storage is insignificant because of the emergence of high capacity high speed secondary storage. The framework only occupies approximately 6.33MB, representing less than 1% of the total secondary storage which goes up to approximately 1GB. Most of the 6.33MB is occupied by the system libraries that can be shared with other resident applications as well.

The program file is only 57KB and the main memory used by the framework while it is running is only approximately 144 KB on a device that has 64MB of main memory. This represents less than 1% usage most of the time while the 4 main threads sleep, though more memory is used when events occur, for example when the backlight must be updated. Below is a Task Manager utility output showing memory consumption:



Process [25]	Mem Usage	Threads
cprog.exe	636 kb	13
SAPSettings.exe	1 kb	2
connmgr.exe	90 kb	13
services.exe	136 kb	32
shell32.exe	774 kb	22
gwes.exe	3655 kb	19
device.exe	2470 kb	143
filesys.exe	1864 kb	17
NK.EXE	1 kb	2
ContextSensiti...	144 kb	4

Figure 31: Main Memory Usage by the Framework

The figure above shows the "ContextSensitive.exe" application (the bottom most process) using only 144kb of main memory. This typically increases to about 800kb when some threads wake up. The framework's memory requirements are reasonable, but care was taken to pre-load data from operations that may take long. An example is the XML rule repository, where all the rules are loaded into a memory buffer in order to read the rules with speed when an event occurs.

Battery Consumption

The consumption of the battery, taken over a 5 day week, is presented in the table below:

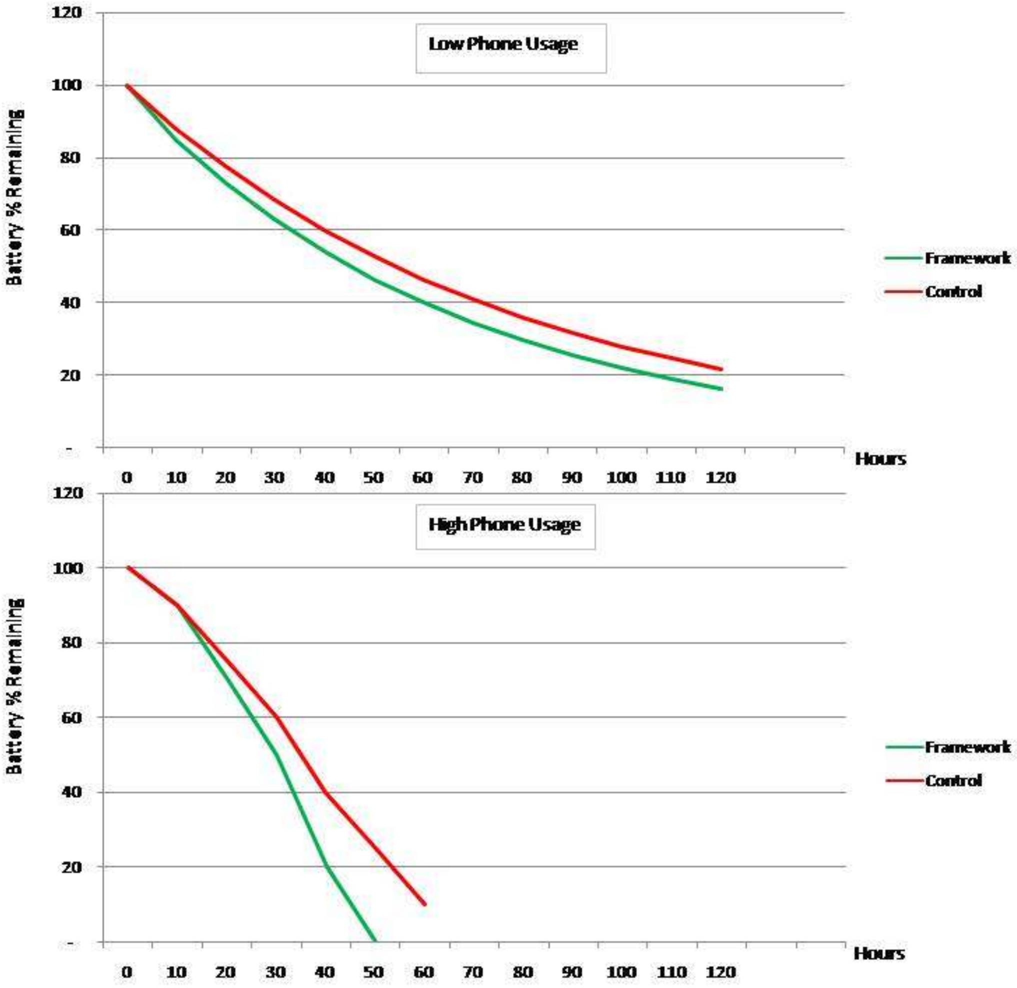


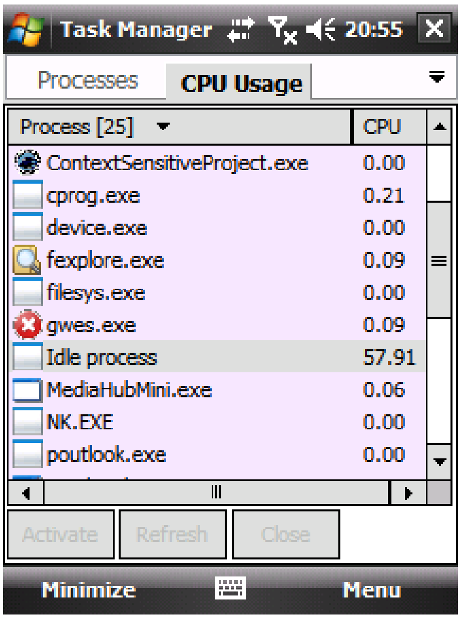
Figure 32: Results of battery consumption

The battery consumption was logged in a ”battery.log” file over a 120 hour period and the results are shown in the line graphs above. The battery level steadily falls for both mobile phones while maintaining a marginal difference. The lower graph shows somewhat higher battery consumption by the framework than the control when the phone usage is high and the battery goes flat within 3 days for both phones. The two phones were subjected to the same environments and calls and messages were sent uniformly over the 120 hour period. The higher battery consumption for a high usage phone is because the framework continues to check for ambient light or noise when the phone is not idle. For all practical intends purposes, the battery life differences are negligible.

This is especially an important finding because previous systems have been based on special batteries [38], but even then, the usage was still somewhat less desirable than mine. The advantages of using .NET’s power-saving features and coding techniques were instrumental.

CPU Time

The diagram shows the framework application with zero CPU usage at its idle time:



Process [25]	CPU
ContextSensitiveProject.exe	0.00
cprog.exe	0.21
device.exe	0.00
fexplore.exe	0.09
filesys.exe	0.00
gwes.exe	0.09
Idle process	57.91
MediaHubMini.exe	0.06
NK.EXE	0.00
poutlook.exe	0.00

Figure 33: CPU usage when framework is "sleeping"

As shown in the figure above, when the framework application is “sleeping” there is no CPU time slice allocated to it. The framework application will only receive CPU slots when one or more of the threads wake up. The framework results in a delay of about a second or two before the alert output as compared to a normal phone without the framework. The delay is caused by the need to check for ambient noise and other context variables.

To have an idea of the effect of the framework on other applications, I designed a small program that executes time stamps using a simple recursive algorithm. The normal execution time for the program is 25 seconds and below is tabulated the execution time when the framework is running and certain events occur:

Table 14: Framework's effect on the phone's speed

Test Description	Framework (seconds)	Control (seconds)
Normal Execution Time		25.000
During a call	28.749	26.345
At SMS delivery	26.985	25.512
During suspend mode	25.100	25.092
During ON mode	26.210	25.135

The table shows the most significant differences when an incoming call is in progress. This is because the framework checks the states of context variables and gives priority to handling of the event than to the time-stamping program. In other circumstances, the framework does not result in a significant difference from the control mobile phone. It is important to note that the time is measured in seconds and milliseconds and the delays may be hardly noticeable.

7.4 Summary

This chapter discusses the results of the entire research. It starts by presenting the results of the survey carried out on 11 diabetes patients then goes on to the controlled experiments. The results clearly show how the context-sensitive framework is useful for creating an intelligent environment that benefits applications on patient-operated mobile terminals.

In the last sections, the effect of the framework on the mobile phone resources is carefully analysed. This is a very important part because it determines whether the framework will be successful on any ordinary mobile phone. The positive results are testimony of the success of the context modelling approach and programming techniques used.

CHAPTER 8

Concluding Remarks and Future Work

8.1 Conclusion

The thesis main problem was to construct a framework for enhancing usability of a diabetes self-help tool through creation of an intelligent environment, based on context information, within which the tool would operate. A major challenge was representation and modelling of the context information, but a solution was developed based on an abstraction method comprising simple deterministic finite state automata. This solution provided a low time and space computation complexity suited to ordinary mobile phones. The developed framework is based on a resource-efficient event-driven architecture. The use of a completely separate rules repository provides for easy customisability of applications based on the framework. In the next paragraphs, inferences are made about the significance of these findings vis-à-vis the original sub-problems.

Question 1: *What context information sources are easily available on mobile phones and are also relevant to managing Type 2 diabetes?*
The process of creatively exploiting context information from sources easily available on the phone is clearly described in the requirements specification chapter. Many context variables were identified and named, including rejected ones. The chosen context variables helped detect some useful contexts.

Question 2: *How can context information be extracted and modelled using generalizable abstractions?*
The design chapter discusses the extraction methods and explains why heterogeneous context information must be abstracted in a generalizable way. A context modelling method is proposed in the same chapter and demonstrations are given about how the abstraction is generalizable, extensible and reusable.

Question 3: *How can the modelled context be used to adapt the self-help tool and the terminal's communication services to changing context?*
An even-driven architecture was proposed for applying rules to the modelled context in order to adapt the phone as described in the design chapter. The architecture is especially efficient because the phone's services and applications are only adapted when an event occurs. This means resources are only used when the need arises, as opposed to other mechanisms such as polling which tend to waste precious resources. The architecture worked well as evidenced by the accurate context detections documented in the discussion chapter.

Question 4: What are the design considerations for a context-sensitive framework on mobile terminals?
The mobile phone is hard pressed for resources and it is imperative to consider this fact when designing applications for mobile phones. The phone is almost always struggling to keep abreast with the core applications already running. The key resources were identified right from the requirements chapter as memory, CPU time and battery life, and other general and irrelevant design considerations were precluded right from the start.
The experiment results were consistent with the original goal of optimizing resource consumption.

Question 5: How can the framework be used to enhance Type 2 diabetes self-help tools?
I describe enhancements that can be applied to a self-help tool that makes it more adaptable to changing context. The enhancements are clearly articulated in the design chapter and implementation chapters and are exemplified using the diabetes self-help tool developed by the research group at NST

8.2 Thesis Contribution

1. Novel context modelling and abstraction method (based on Question2 sub-problem)
The thesis proposes modelling context information using a generalizable, reusable and extensible abstraction comprising 2-state finite state automata. The method is suitable for resource-starved environments and provides context states as services accessed through the exported Tag/Value pair interface, thus the information can be shared with other applications easily.

Context modelling is still a great challenge that has inhibited development of context-sensitive applications. The proposed generalizable abstraction presents an opportunity for standardization and further development of context-sensitive applications on mobile phones.

2. Demonstrated the efficiency of the event-driven architectures for adaptation (based on Question3 sub-problem)
The thesis shows how the Event-Condition-Action (ECA) paradigm built on the event-driven architecture is a concrete approach for using context information to adapt mobile applications efficiently. In addition, the thesis demonstrates how this approach provides an efficient method for handling events in environments where context changes rapidly while resources are limited.

3. Separation of rules from logic (again, based on Question3 sub-problem)
The prescribed true separation of the inference engine from the rules through the use of an independent XML-based rule repository presents an opportunity for a highly customizable maintainable context-sensitive application based on the framework.

4. Enhanced usability and disease management (Based on Question5 sub-problem)
The context-sensitive framework significantly improves usability of self-help tools on multipurpose mobile terminals. The proposed enhancements of self-help tools, such as message themes, visual and sound effects make the tools more appealing and easily adaptable to changing contexts.

This research also enhances our knowledge about user preferences when it comes to developing context-sensitive patient-operated disease management applications, especially for diabetes patients past their middle age.

5. Introduced creative and innovative ways of identifying and extracting context information from otherwise unlikely sources. (Based on Question1 sub-problem)
The framework is based on innovative ways of extracting context information from otherwise unlikely sources resident on a mobile phone. The thesis therefore stimulates creativity and critical thinking necessary for constructing modern context-sensitive systems. This is an important contribution because specialised sensors may not be widely available on ordinary mobile phones in the foreseeable future.

6. Reusable objects and components
The developed objects and components are reusable and are based on minimal resource requirements, thus they are easy to adapt for several types of mobile phones. The developed components are open, can be modified and some useful methods are developed in C++ in order to increase portability and the support base.

8.3 Future Work

1. Capturing User Preferences and Behaviour to Reduce Uncertainties
In this thesis I have not dealt adequately with uncertainties because the major goal was to be able to process context information efficiently and accurately. In the future, possible uncertainties can be dealt with by considering user experiences and behaviour over time and using the information to mould the rules dynamically. This is because user preferences may change and the user should not need to reconfigure the behaviour of an application, otherwise autonomy of the applications based on my framework will be virtually lost.

2. Standardizing Context Tag Classes ("Tags" as described in the design chapter)
It would be useful in the future to have a complete reference guide for flexible context variables tag class naming. This thesis only gave a few examples based on the used context variables. In the future, a standard can be developed for all the possible tag classes, based on a comprehensive context information classification method, some of which were discussed in the Theoretical Framework chapter. For example, a tag class could be "Ambient Light", but extensibility means the developers have the flexibility to add tags to the class such as "Fluorescent Light", "Neon Light" or "Sun Light".

3. Easy Rule Configurations

The current XML rules repository is based on manual and elaborate logical rule deductions. This is a crucial step that, if it goes wrong, the phone may behave strangely. Therefore, a tool can be developed to deduce the rules in an easy way. For example, the tool can have a web interface where the user can enter, modify or delete rules. The tool would then deduce the final rule-set automatically.

4. Longitudinal Field Trials

Early clinical trials have already proven that SMBG augmented with some education does result in good glycaemic control. Longitudinal trials will be required to determine whether increasing usability of self-help tools affects the patients' motivation to keep using the tools over protracted periods of time.

If usability has a direct relationship with the motivation, then using context-sensitivity to improve usability of mobile terminals would be a highly significant area for further research. My initial feelings are that there are many factors that influence a patient to either continue or withdraw use of the self-help tools and lack of usability is only one of them.

APPENDICES

Appendix-1A: Sorted Rules Matrix

Noise	Activity	Calendar	Time	SMS/Popup	Calls
0	0	0	0	Beep	
1	0	0	0	Beep	
1	0	0	1	Beep	
0	1	0	0	Beep + Vibration	
0	1	0	1	Beep + Vibration	
1	1	0	0	Beep + Vibration	
1	1	0	1	Beep + Vibration	
1	1	1	0	Beep + Vibration	
1	1	1	1	Beep + Vibration	
0	0	0	1	LED Flash	
0	0	1	0	Vibration	
0	0	1	1	Vibration	
0	1	1	0	Vibration	
0	1	1	1	Vibration	
1	0	1	0	Vibration	
1	0	1	1	Vibration	
0	0	0	1		Beep + LED Flash
0	0	0	0		Ring
1	0	0	0		Ring
1	0	0	1		Ring + LED Flash
0	1	0	0		Ring + Vibration
0	1	0	1		Ring + Vibration
1	1	0	0		Ring + Vibration
1	1	0	1		Ring + Vibration
0	0	1	0		Vibration
0	0	1	1		Vibration
0	1	1	0		Vibration
0	1	1	1		Vibration
1	0	1	0		Vibration
1	0	1	1		Vibration
1	1	1	0		Vibration
1	1	1	1		Vibration

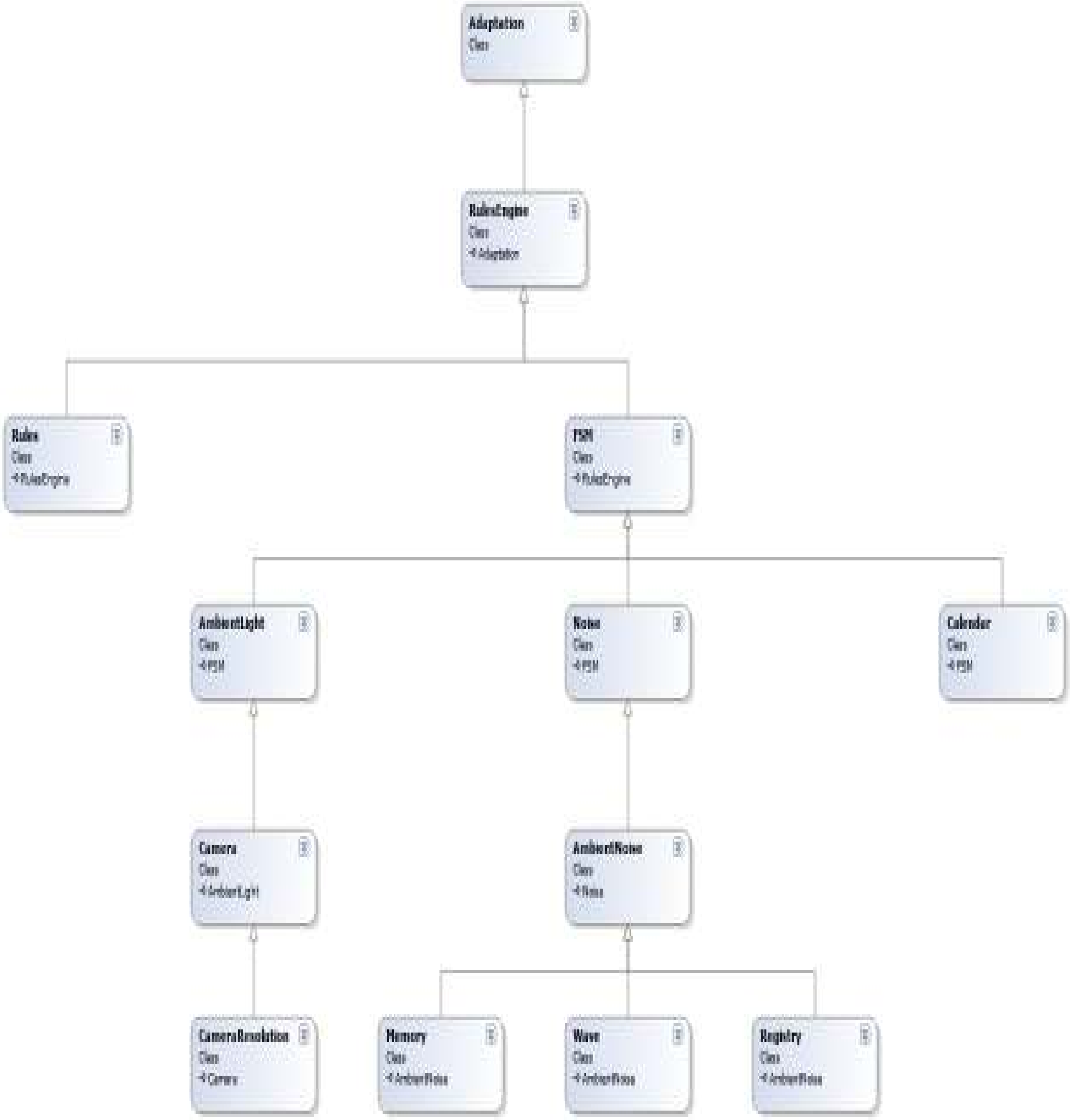
Appendix-1B: Raw Accept & Deny Rules Matrix

	Accept rules				
Noise	**Activity**	**Calendar**	**Time**	**SMS/Popup**	**Call**
0	0	0	0	Beep	Ring
0	0	0	1	LED Flash	Beep + LED Flash
0	0	1	0	Vibration	Vibration
0	0	1	1	Vibration	Vibration
0	1	0	0	Beep + Vibration	Ring + Vibration
0	1	0	1	Beep + Vibration	Ring + Vibration
0	1	1	0	Vibration	Vibration
0	1	1	1	Vibration	Vibration
1	0	0	0	Beep	Ring
1	0	0	1	Beep	Ring + LED Flash
1	0	1	0	Vibration	Vibration
1	0	1	1	Vibration	Vibration
1	1	0	0	Beep + Vibration	Ring + Vibration
1	1	0	1	Beep + Vibration	Ring + Vibration
1	1	1	0	Beep + Vibration	Vibration
1	1	1	1	Beep + Vibration	Vibration
	Deny Rules				
Priority	**Noise Level**	**Calendar Priority**	**Time**	**SMS/Popup**	**Call**
0	0	0	1	Alert deferred + Notify sender	Voicemail
0	0	1	0	Alert deferred + Notify sender	Voicemail
0	0	1	1	Alert deferred + Notify sender	Voicemail
0	1	0	0	Alert deferred + Notify sender	Voicemail
0	1	0	1	Alert deferred + Notify sender	Voicemail
0	1	1	0	Alert deferred + Notify sender	Voicemail
0	1	1	1	Alert deferred + Notify sender	Voicemail
1	0	1	0	Alert deferred + Notify sender	Voicemail
1	0	1	1	Alert deferred + Notify sender	Voicemail
1	1	1	0	Alert deferred + Notify sender	Voicemail
1	1	1	1	Alert deferred + Notify sender	Voicemail

Appendix-2A: Rule file XML schema definition

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="rules" targetNamespace="http://tempuri.org/rules.xsd"
elementFormDefault="qualified" xmlns="http://tempuri.org/rules.xsd"
xmlns:mstns="http://tempuri.org/rules.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="RuleSet">
    <xs:sequence>
      <xs:element name="Deny">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Rule">
              <xs:complexType>
                <xs:sequence>
                </xs:sequence>
                <xs:attribute name="condition" type="xs:string" />
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Allow">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Rule">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="condition">
                    <xs:complexType>
                      <xs:sequence>
                      </xs:sequence>
                      <xs:attribute name="string" type="xs:string" />
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="action">
                    <xs:complexType>
                      <xs:sequence>
                      </xs:sequence>
                      <xs:attribute name="call" type="xs:string" />
                      <xs:attribute name="sms" type="xs:string" />
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Appendix-3A: Class Diagram



Appendix-4A: Survey Questionnaire

Spørreskjema April 2008 – Studentoppgave brukervennlighet

1. I hvilken aldersgruppe er du?

<45 □ 46-55 □ 56-65 □ 66-75 □

2. Har du brukt mobiltelefon lenger enn 2 år?

Ja □ Nei □

3. Har du brukt noe elektronisk sykdoms-hjelpemiddel før?

Ja □ Nei □ Hvis ja, oppgi noen detaljer: ______________________________

4. Bruker du å angi påminnelser for avtaler/oppgaver på din mobiltelefon / PC?

Ja □ Nei □

5. Hva gjør du når du trimmer?

Går/jobber □ Gym □ Husarbeid/hagearbeid □

Annet (Spesifiser):_____________________________________

6. Kan du rangere følgende ubehagelige situasjoner. Angi tall, der 1 = mest ubehagelig.

□ Mobiltelefonen ringer høyt når du er i et viktig møte eller samtale.

□ Tap av en viktig telefonsamtale fordi du ikke hørte ringingen.

□ Motta en oppringing på en støyfull plass der du knapt kan høre den som prater i telefonen.

□ Gå tom for strøm på telefonen når du ikke har med deg laderen eller tilgang til et strøm-støpsel.

□ Motta en feiloppringing når du ligger å sover klokka 3 på natta.

□ Lese SMS i mørket og displayet på telefonen er veldig lyst.

7. Hvilke forbedringer ville du ønsket deg på mobiltelefoner?

____

Tusen Takk!!

Appendix-4B: Survey Questionnaire English Translation

This short survey is part of an MSc research project.

1. What is your age range?

<45 □ 46-55 □ 56-65 □ 66-75 □ >76 □

2. Do you have more than 2 years experience using a mobile device?

Yes □ No □

3. Have you used any electronic disease management system before?

Yes □ No □ If yes, please give details: ______________________________

4. Do you schedule reminders for appointments/tasks on your mobile/PC?

Yes □ No □

5. What do you do when you exercise?

Walk/Jog □ Gym □ House chores/Gardening □

Other (Specify):____________________________________

6. Please rank the following unpleasant situations. Assign numbers 1-5, where 1 = most unpleasant.

□ The mobile phone rings loudly when you are in an important meeting/conversation

□ Missing an important phone call because you didn’t hear the phone ring

□ Getting a call in a noisy place when you can hardly hear the other person on the line

□ Running out of battery when you do not have the charger or a power outlet

□ Getting a stray (wrong number) phone call when you are asleep at 3AM

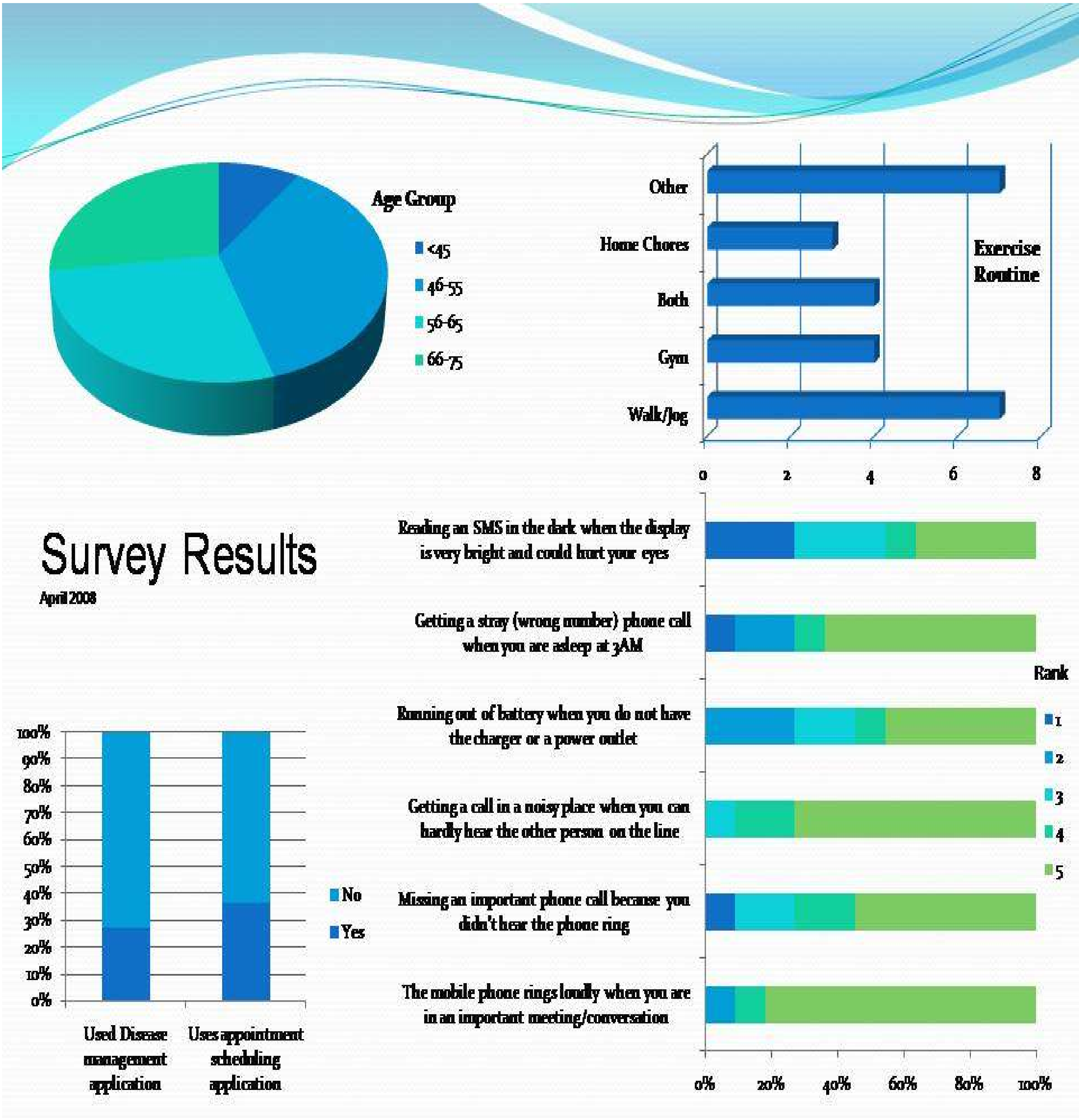
□ Reading an SMS in the dark when the display is very bright and could hurt your eyes

7. What improvements would you like to see on mobile devices?

____

Thank You!!

Appendix-4C: Survey Results Flyer



Survey Results

April 2008

References

1. Hornbæk, K. and E.L.-C. Law, *Meta-analysis of correlations among usability measures*, in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2007, ACM: San Jose, California, USA.

2. Årsand E., Varmedal R., Hartvigsen G. , *Usability of a Mobile Self-Help Tool for People with Diabetes: the Easy Health Diary.* IEEE Conference on Automation Science and Engineering. Arizona, USA, 2007. **2007**(1): p. 863-868.

3. Årsand E., Andersson N., Hartvigsen G., *No-Touch Wireless Transfer of Blood Glucose Sensor Data. .* COGnitive systems with Interactive Sensors 2007. California, USA, 2007. **2007**(1).

4. Årsand E., Olsen O. A., Varmedal R., Mortensen W., Hartvigsen G., *A System for Monitoring Physical Activity Data Among People with Type 2 Diabetes.* The 21st International Congress of the European Federation for Medical Informatics. Göteborg, Sweden, 2008. **2008**(1).

5. Årsand E., Tufano J.T., Ralston J., Hjortdahl P. , *Designing Mobile Dietary Management Support Technologies for People with Diabetes.* Tromsø Telemedicine and eHealth Conference. Tromsø, Norway., 2008. **2008**(1).

6. Yoon, K.H. and H.S. Kim, *A short message service by cellular phone in type 2 diabetic patients for 12 months.* Diabetes Res Clin Pract, 2008. **79**(2): p. 256-61.

7. Benjamin, E.M., *Self-Monitoring of Blood Glucose: The Basics.* Clinical Diabetes, 2002. **20** (1): p. 45-47.

8. Davis, W.A., D.G. Bruce, and T.M. Davis, *Is self-monitoring of blood glucose appropriate for all type 2 diabetic patients? The Fremantle Diabetes Study.* Diabetes Care, 2006. **29**(8): p. 1764-70.

9. Welschen, L.M., et al., *Self-monitoring of blood glucose in patients with type 2 diabetes who are not using insulin.* Cochrane Database Syst Rev, 2005(2): p. CD005060.

10. Kerkenbush, N.L. and C.E. Lasome, *The emerging role of electronic diaries in the management of diabetes mellitus.* AACN Clin Issues, 2003. **14**(3): p. 371-8.

11. Ferrer-Roca, O., et al., *Web-based diabetes control.* J Telemed Telecare, 2004. **10**(5): p. 277-81.

12. Kim, C.J. and D.H. Kang, *Utility of a Web-based intervention for individuals with type 2 diabetes: the impact on physical activity levels and glycemic control.* Comput Inform Nurs, 2006. **24**(6): p. 337-45.

13. McMahon G.T., Gomes H.E., Hickson-Hohne S., Hu T.M.J., Levine B.A., Conlin P.R., *Web-based care management in patients with poorly controlled diabetes mellitus. .* Diabetes Care, 2005(28): p. 1624-9. .

14. Wu C., Steinbaur J.R., Kuo G.M. , *Development of a Web-based diabetes patient management tool.* . AMIA Annu Symp Proc., 2005. **1157**.

15. Peel, E., M. Douglas, and J. Lawton, *Self monitoring of blood glucose in type 2 diabetes: longitudinal qualitative study of patients' perspectives.* BMJ, 2007. **335**(7618): p. 493.

16. Keshavjee, K., et al., *Technology failure analysis: understanding why a diabetes management tool developed for a Personal Digital Assistant (PDA) didn't work in a randomized controlled trial.* AMIA Annu Symp Proc, 2003: p. 889.

17. Denning, P.J., et al., *Computing as a discipline.* Commun. ACM, 1989. **32**(1): p. 9-23.

18. Abowd G.D., Dey A.K., Brown P.J., Davies N., Smith M., Steggles P., *Towards a Better Understanding of Context and Context-Awareness*, in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. 1999, Springer-Verlag: Karlsruhe, Germany.

19. Collins, H., *Collins Essential English Dictionary*. 2006, HarperCollins Publishers.

20. Want, R., et al., *The active badge location system.* ACM Trans. Inf. Syst., 1992. **10**(1): p. 91-102.

21. Schmidt A., Beigl B., Gellersen H-W. , *There is more to Context than Location.* Computers & Graphics Journal, Elsevier, 1999. **23**(6): p. 893-902.

22. Schilit B., Adams N., Want R., *Context-Aware Computing Applications.* IEEE Computer Society, In Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA,, 1994. **1994**(1): p. 85-90.

23. Brown, P.J., *The Stick-e Document: a Framework for Creating Context-Aware Applications.* . Electronic Publishing ’96, 1996. **1996**(1): p. 259-272.

24. Dey, A.K., *Context-Aware Computing: The CyberDesk Project.* . AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report 1998. **1998**(SS-98-02): p. 51-54.

25. Dey A.K., Abowd G.D., *Toward a better understanding of context and context-awareness.* GVU Technical Report, College of Computing, Georgia Institute of Technology., 1999. **1999**(GIT-GVU-99-22).

26. Pascoe, J., *Adding Generic Contextual Capabilities to Wearable Computers*, in *Proceedings of the 2nd IEEE International Symposium on Wearable Computers*. 1998, IEEE Computer Society.

27. Asim Smailagic, et al., *eWatch: Context Sensitive System Design Case Study*, in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design*. 2005, IEEE Computer Society.

28. Mika Raento, et al., *ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications.* IEEE Pervasive Computing, 2005. **4**(2): p. 51-59.

29. Nokia. *Activity Monitor*. Nokia Research Center 2007 [cited; Available from: http://research.nokia.com/projects/activity_monitor.

30. Kaenampornpan M., O'Neill E., Kostakos V., Warr A., *Classifying Context Classifications: an Activity Theory Perspective. .* 2nd UK-UbiNet Workshop, University of Cambridge, UK, 2004.

31. Korkea-Aho, M., *Context-aware applications survey.* Internetworking Seminar, Helsinki University of Technology 2000. **Tik-110.551**(1).

32. Pascoe, J., *The stick-e note architecture: extending the interface beyond the user*, in *Proceedings of the 2nd international conference on Intelligent user interfaces*. 1997, ACM: Orlando, Florida, United States.

33. Arun, M., *SmartReminder: A Case Study on Context-Sensitive Applications*. 2001, Dartmouth College.

34. Froehlich, J., et al., *MyExperience: a system for <i>in situ</i> tracing and capturing of user feedback on mobile phones*, in *Proceedings of the 5th international conference on Mobile systems, applications and services*. 2007, ACM: San Juan, Puerto Rico.

35. Daniel Siewiorek, et al., *SenSay: A Context-Aware Mobile Phone*, in *Proceedings of the 7th IEEE International Symposium on Wearable Computers*. 2003, IEEE Computer Society.

36. Krause A., et al., *Unsupervised, Dynamic Identification of Physiological and Activity Context in Wearable Computing*, in *Proceedings of the 7th IEEE International Symposium on Wearable Computers*. 2003, IEEE Computer Society.

37. Korhonen I., Paavilainen P., Särelä A., *Application of ubiquitous computing technologies for support of independent living of the elderly in real life settings.* In: UbiHealth 2003: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications 2003, 2003. **2003**(1).

38. Asim, S., et al., *eWatch: Context Sensitive System Design Case Study*, in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design (ISVLSI'05) - Volume 00*. 2005, IEEE Computer Society.

39. Bellotti V., Edwards K., *Intelligibility and accountability: Human considerations in context-aware systems.* Human-Computer Interaction, 2001. **16**(2-4): p. 193-212.

40. Mäntyjärvi J., Tuomela U., Känsälä I., Häkkilä J. , *Context Studio – Tool for Personalizing Context-Aware Application in Mobile Terminals. .* Proc. OZCHI, 2003. **2003**(1): p. 64-73.

41. Barkhuus L., Dey A., *Is Context-Awareness Taking Control Away from the User? Three Levels of Interactivity Examined.* In Proceedings of Ubicomp, 2003. **2003**(1): p. 159-166.

42. Jonna, H., et al., *Developing design guidelines for context-aware mobile applications*, in *Proceedings of the 3rd international conference on Mobile technology, applications \&amp; systems*. 2006, ACM: Bangkok, Thailand.

43. Jaime M., Serrano O., Joan Serrat F., Kun Yang, Epi Salamanca C., *Modelling Context Information for Managing Pervasive Network Services.* Proceedings of ICMS' 05, Marrakech, Morocco, 2005.

44. Henricksen Karen, Indulska Jadwiga, Rakotonirainy Andry *Modeling Context Information In Pervasive Computing Systems .* In Proceedings Pervasive, Zurich, Switzerland., 2002. **02**: p. 79-117.

45. Patrick, B., zillon, and C. Marcos, *Modeling and using context.* Knowl. Eng. Rev., 1998. **13**(2): p. 185-194.

46. Kaenampornpan, M., , O'Neill, E., *Integrating History and Activity Theory in Context Aware System Design., 2005 8-13 May, 2005,.* In proceedings of the W8 ECHISE 2005 - 1st International Workshop on Exploiting Context Histories in Smart Environments, Munich, Germany., 2005. **Pervasive 2005**.

47. Li, R., R. Li, and J. Wen, *A Ptolemy II Based Modeling Approach for Context-Aware Computing.* IEEE, 1st International Symposium on Pervasive Computing and Applications, 2006.

48. Loke, S., *Context-Aware Artifacts: Two Development Approaches.* IEEE Pervasive Computing, 2006. **5**(2): p. 48-53.

49. Brdiczka, O., et al., *Deterministic and Probabilistic Implementation of Context*, in *Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*. 2006, IEEE Computer Society.

50. Mokhta, S.B., et al., *Context-Aware Service Composition in Pervasive Computing Environments* Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2006. **3943/2006**: p. 129-144.

51. Dourish, P., *What we talk about when we talk about context.* Personal and Ubiquitous Computing, 2004. **8**(19-30).

52. Engeström, Y., , Miettinen, R., Punamäki, R.L., *Perspectives on Activity Theory.* , ed. C.U. Press. 1999.

53. Payton J., Roman C., Julien C., *"Context-Sensitive Data Structures Supporting Software Development in Ad Hoc Mobile Settings"* Proceedings of the 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems, co-located with ICSE 2004., 2004. **2004**(1).

54. Flora C. D., Riva O., Raatikainen K., Russo S. , *Supporting mobile context-aware applications through a modular service infrastructure. .* In Poster Proceedings of 6th International UbiComp 2004 Conference, 2004. **2004**(1): p. 7-10.

55. Coheso. *Personal Health Management Tools* 2007 [cited; Available from: http://www.coheso.com/diabetesdiary001.html.

56. N Zhao, A Roudsari, E Carson, *A Web-based Diabetes Management System: DiabNet.* J Med Internet Res, 1999. **1**(1).

57. Mahesh, S., et al., *Healthcare@Home: Research Models for Patient-Centred Healthcare Services*, in *Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing*. 2006, IEEE Computer Society.

58. Hall, D.L., et al., *A Web-based interprofessional diabetes education course.* Am J Pharm Educ, 2007. **71**(5): p. 93.

59. Kim, C., et al., *Internet diabetic patient management using a short messaging service automatically produced by a knowledge matrix system.* Diabetes Care, 2007. **30**(11): p. 2857-8.

60. Kim, H.S., *A randomized controlled trial of a nurse short-message service by cellular phone for people with diabetes.* Int J Nurs Stud, 2007. **44**(5): p. 687-92.

61. Kim, S.I. and H.S. Kim, *Effectiveness of mobile and internet intervention in patients with obese type 2 diabetes.* Int J Med Inform, 2007.

62. Walseth O. A., Arsand E., Sund T., Skipenes E., *Wireless transfer of sensor data into electronic health records.* Stud Health Technol Inform, 2005. **116**: p. 334-9.

63. Cocosila, M. and N. Archer, *A framework for mobile healthcare answers to chronically ill outpatient non-adherence.* Inform Prim Care, 2005. **13**(2): p. 145-52.

64. Cocosila, M., C. Coursaris, and Y. Yuan, *M-healthcare for patient self-management: a case for diabetics.* Int J Electron Healthc, 2004. **1**(2): p. 221-41.

65. Hung, S.H., et al., *Care for Asthma via Mobile Phone (CAMP).* Stud Health Technol Inform, 2007. **126**: p. 137-43.

66. Anhoj, J. and C. Moldrup, *Feasibility of collecting diary data from asthma patients through mobile phones and SMS (short message service): response rate analysis and focus group evaluation from a pilot study.* J Med Internet Res, 2004. **6**(4): p. e42.

67. Morak, J., et al., *Improving telemonitoring of heart failure patients with NFC technology*, in *Proceedings of the fifth IASTED International Conference: biomedical engineering*. 2007, ACTA Press: Innsbruck, Austria.

68. Broens, T., et al., *Towards an application framework for context-aware m-health applications.* Int. J. Internet Protoc. Technol., 2007. **2**(2): p. 109-116.

69. Oliver, A., T. Gerhard, and ster, *Recognition of dietary activity events using on-body sensors.* Artif. Intell. Med., 2008. **42**(2): p. 121-136.

70. Kempf, K., et al., *Self-monitoring of blood glucose in type 2 diabetes: a new look at published trials.* Diabetologia, 2008.

71. Farmer, A., et al., *Impact of self monitoring of blood glucose in the management of patients with non-insulin treated diabetes: open parallel group randomised trial.* BMJ, 2007. **335**(7611): p. 132.

72. McAndrew, L., et al., *Does patient blood glucose monitoring improve diabetes control? A systematic review of the literature.* Diabetes Educ, 2007. **33**(6): p. 991-1011; discussion 1012-3.

73. O'Kane, M.J., et al., *Efficacy of self monitoring of blood glucose in patients with newly diagnosed type 2 diabetes (ESMON study): randomised controlled trial.* BMJ, 2008. **336**(7654): p. 1174-7.

74. Årsand E., Hartvigsen G. , *A wearable eHealth system for people with Type 2 diabetes.* Scandinavian conference on Health Informatics. Aalborg, Denmark, 2005. **2005**(1).

75. James Robertson, Suzanne Robertson. *Volere Requirements Specification Template*. 2006 [cited 11; Available from: www.volere.co.uk.

76. Mitchell, C. *Adjust Your Ring Volume For Ambient Noise.* 2007 [cited; Available from: http://msdn.microsoft.com/msdnmag/issues/07/10/NoiseDetection/default.aspx.

77. Microsoft. *Multiple Channel Audio Data and WAVE Files*. 2007 [cited; Available from: http://www.microsoft.com/whdc/device/audio/multichaud.mspx#EWCAC.

78. Reiter, C., *With J: image processing 2: color spaces.* SIGAPL APL Quote Quad, 2004. **34**(3): p. 3-12.

79. Thomas Muhl, Marcel Binnebosel, Uwe Klinge, Thomas Goedderz, *New Objective Measurement to Characterize the Porosity of Textile Implants.* Journal of Biomedical Materials Research Part B: Applied Biomaterials, 2007. **84B**(1): p. 176-183.

80. Jovanov, E., et al., *A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation.* J Neuroeng Rehabil, 2005. **2**(1): p. 6.

81. Anind K. Dey, Daniel Salber, Gregory D. Abowd *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications* Anchor article of a special issue on context-aware computing in the Human-Computer Interaction (HCI) Journal, 2001. **16**(2-4): p. 97-166.

82. Wagner, F., *Modeling Software with Finite State Machines: A Practical Approach*. Vol. ISBN 0-8493-8086-3. 2006: Auerbach Publications.

83. Hopcroft, J.E. and J.D. Ullman, *Introduction To Automata Theory, Languages, And Computation*. 1990: Addison-Wesley Longman Publishing Co., Inc. 418.

84. Harmon P., Sawyer B., *Creating Expert Systems for Business and Industry*. Vol. ISBN 0471614955. 1990: Wiley.

85. Loke, S., *Context-Aware Pervasive Systems: Architectures for a New Breed of Applications*. 2007: Auerbach Publications.

86. Alferes J.J., Banti F., Brogi A. , *An Event-Condition-Action Logic Programming Language.* Logics in Artificial Intelligence, Springer Berlin / Heidelberg, 2006. **ISBN 978-3-540-39625-3**.

87. Microsoft. *Platform Invoke Tutorial*. 2008 [cited; Available from: http://msdn2.microsoft.com/en-us/library/aa288468.aspx.

88. Treasury, O.o.t.U.K.s. 1995 [cited; Available from: http://www.comp.glam.ac.uk/pages/staff/tdhutchings/chapter4.html.

89. Raffman, A. *The Wavedev2 Gainclass Implementation*. Windows CE Multimedia Team Blog 2007 [cited 2007 1]; Available from: http://blogs.msdn.com/medmedia/archive/2007/01/04/the-wavedev2-gainclass-implementation.aspx.