TeleKids - Mobile application for kids with Down syndrome

Amir Erfani, Johan Ho

TDT4735 Software Engineering, Depth Study

Supervisor: Steinar Brede
Subject teacher: Heri Ramampiaro

Trondheim, 14 December 2005

NTNU
telenor
Abstract

This report describes our project, where we have studied the use of scanning technology on a cellphone device, to enable children with Down syndrome to use a cellphone. We have researched different scanning technologies that are possible to use with a mobile device, and studied and chosen other technologies and functionalities that we want to add in our application. An architecture was made using the technologies and functionality we’ve chosen, and a application was developed.

The goal of this project was to create an application with our chosen technologies and see if it is a help for children with Down syndrome, and the main results of this project is a description of how well our project has fulfilled this goal.
b Preface

This report is the result of the work done by Amir Erfani and Johan Ho in the subject “TDT4735 Software Engineering, Depth Study” at the Institute for Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU) Autumn 2005.

We would like to thank our supervisor, Steinar Brede from Telenor Research & Development, for providing us with resources, ideas and feedback on our project. We would also like to thank our subject teacher Heri Ramampiaro, for giving feedback and tips for our report.

Trondheim, 14 December 2005

_________________________________________  _______________________________________
Amir Erfani                                        Johan Ho
c Contents

a Abstract ................................................................................................................................. 3
b Preface .................................................................................................................................. 5
1 Introduction .......................................................................................................................... 11
   1.1 Background ...................................................................................................................... 11
   1.2 Motivation ......................................................................................................................... 11
   1.3 User scenarios ................................................................................................................... 12
      1.3.1 Scenario 1: Communication with parents from school ............................................. 12
      1.3.2 Scenario 2: Communication with friends from home ................................................. 13
      1.3.3 Scenario 3: Configuration of the application ................................................................. 13
   1.4 Project context .................................................................................................................. 13
   1.5 Research methodology .................................................................................................... 14
   1.6 Outline of the Report ....................................................................................................... 15
2 Prestudy .................................................................................................................................. 16
   2.1 Related work ...................................................................................................................... 16
      2.1.1 EiT project [2] ............................................................................................................... 16
      2.1.2 Plant scanner [3] .......................................................................................................... 18
   2.2 Down syndrome ............................................................................................................... 19
      2.2.1 Research from written material .................................................................................. 19
      2.2.2 Information from parents ............................................................................................ 20
   2.3 Context .............................................................................................................................. 20
   2.4 Mobile phones ................................................................................................................. 21
      2.4.1 Qtek s100. .................................................................................................................... 21
      2.4.2 Nokia 5140 .................................................................................................................. 22
   2.5 Programming languages and development environments ................................................. 22
      2.5.1 Windows CE ............................................................................................................... 22
      2.5.2 Visual C++ for Windows CE ......................................................................................... 23
      2.5.3 Compact Framework .NET [12] ................................................................................... 23
      2.5.4 Series 40 (Nokia) ....................................................................................................... 24
      2.5.5 Java 2 Micro Edition (J2ME) ......................................................................................... 24
   2.6 Technologies for scanning data tags .................................................................................... 25
      2.6.1 RFID ........................................................................................................................... 25
      2.6.2 Semacode [13] ............................................................................................................. 27
2.6.3 Barcode reader............................................................................................................ 28
2.7 Technologies for functionality .......................................................................................... 28
  2.7.1 Phone calls, SMS and MMS ......................................................................................... 28
  2.7.2 Text-to-Speech ............................................................................................................ 29
  2.7.3 Push-to-talk ................................................................................................................ 30
  2.7.4 Text-to-video ............................................................................................................. 30
  2.7.5 Text-to-symbols ......................................................................................................... 31
  2.7.6 Camera ....................................................................................................................... 31
2.8 Data storage....................................................................................................................... 32
  2.8.1 XML ........................................................................................................................... 32
  2.8.2 Client-Server .............................................................................................................. 32
2.9 Choices ............................................................................................................................. 33
3 Requirement specification ................................................................................................... 34
  3.1 Functional requirements .................................................................................................. 34
    3.1.1 Use case diagram for users (children) ............................................................... 35
    3.1.2 Requirements for users (children) ....................................................................... 35
    3.1.3 Use case diagram for the configurators(guardians) ......................................... 36
    3.1.4 Requirements for configurators (parents) ........................................................... 36
  3.2 Quality Requirements .................................................................................................... 37
    3.2.1 Usability .................................................................................................................. 37
    3.2.2 Modifiability ........................................................................................................... 38
    3.2.3 Availability ............................................................................................................. 38
    3.2.4 Testability ............................................................................................................... 39
    3.2.5 Performance ............................................................................................................ 39
    3.2.6 Security .................................................................................................................. 40
4 Architecture ....................................................................................................................... 40
  4.1 Architectural drivers ....................................................................................................... 40
  4.2 Stakeholders and concerns ............................................................................................. 40
    4.2.1 Users and acquirers of the system ........................................................................ 40
    4.2.2 Developers of the system ...................................................................................... 41
    4.2.3 Maintainers of the system ..................................................................................... 41
  4.3 Selection of architectural viewpoints ............................................................................. 41
    4.3.1 Logical viewpoint .................................................................................................... 42
    4.3.2 Process view ........................................................................................................... 42
4.3.3 Development view ................................................................. 42
4.4 Quality tactics .................................................................................. 42
  4.4.1 Usability ....................................................................................... 42
  4.4.2 Modifiability ............................................................................... 43
  4.4.3 Testability .................................................................................... 43
  4.4.4 Security ....................................................................................... 43
  4.4.5 Performance ............................................................................... 43
  4.4.6 Availability ................................................................................ 43
4.5 Architectural patterns ........................................................................ 44
  4.5.1 Task Control Architecture Pattern .............................................. 44
4.6 Views .................................................................................................. 44
  4.6.1 Logical view ............................................................................... 44
  4.6.2 Process view ............................................................................... 52
  4.6.3 Development view ....................................................................... 55
5 Development ....................................................................................... 56
  5.1 User mode ....................................................................................... 56
    5.1.1 Sending SMS ........................................................................... 56
    5.1.2 Receiving SMS ......................................................................... 57
    5.1.3 Making phone calls ................................................................... 57
    5.1.4 Receiving phone calls ................................................................. 58
  5.2 Configuration mode ......................................................................... 58
    5.2.1 Entering configuration mode ....................................................... 58
    5.2.2 Contacts .................................................................................. 59
    5.2.3 Actions ..................................................................................... 59
    5.2.4 Options .................................................................................... 60
    5.2.5 ID reader .................................................................................. 61
  5.3 Encountered issues .......................................................................... 61
6 Evaluation ............................................................................................ 62
  6.1 Evaluation of research methodology .............................................. 63
  6.2 Evaluation of the architecture ......................................................... 63
    6.2.1 Functional requirements for the users (children) ......................... 63
    6.2.2 Functional requirements for the parents ....................................... 64
    6.2.3 Quality requirements ................................................................. 65
  6.3 Evaluation of the goals .................................................................... 67
6.3.1 Secondary goal G1 ........................................................................................................ 67
6.3.2 Secondary goal G2 ........................................................................................................ 68
6.3.3 Secondary goal G3 ........................................................................................................ 68
7 Conclusion ............................................................................................................................... 69
8 Further work ............................................................................................................................ 70
9 Glossary ................................................................................................................................... 71
10 References ............................................................................................................................. 73
11 Appendices .............................................................................................................................. 74
  11.1 Specifications of Qtek s100 ............................................................................................... 74
  11.2 Nokia 5140 Technical Specifications ............................................................................... 75
1 Introduction

1.1 Background

According to a study done by “Post og Teletilsynet” in 2005, there are currently more cellphone subscriptions than there are inhabitants in Norway. Even though it just means that a lot of people have more than one subscription, it still shows how that the usage of cellphones has become quite widespread. Still, even with this, very few cellphones have been made to accommodate the needs of people with disabilities.

One group of people that have difficulty learning to use a normal cellphone, are people with Down syndrome. Down syndrome is a term used for a number of genetic disorders which causes a person to have learning difficulties and also physical disabilities. Kids with Down syndrome are therefore generally underdeveloped compared to other kids at the same age. Because of the learning difficulties, these kids have a hard time learning to use technologies like cellphones. At the same time, because of their condition they also require a bit more attention than normal developed kids, and they would have benefited from being able to communicate with their guardians even when they’re not with them.

In an EiT project at NTNU spring 2005, the main goal was to develop a cellphone application to help kids with Down syndrome to communicate with their parents using a cellphone. The application needed to have an easily understandable graphical user interface, which should be easy for the kids to use, and for that purpose a cellphone with a large touch screen was used. However, we heard from those who worked with that project, and from the parents who had tested their application, that there was a problem with their application. Even with a graphical user interface, the kids had problems using it. The reason was that even if the symbols used in the interface was understandable for most people, it’s not a given that the kids with Down syndrome could relate to them.

1.2 Motivation

Since a pure graphical interface wasn’t enough to enable kids with Down syndrome to use a cellphone easily, what other solutions can be tried? We wanted to see if we could develop an application which could enable kids with Down syndrome to use cellphones to communicate with other people.
Another motivation for this project is to try and use technology that hasn’t earlier been applied to personal usage. More specifically, we want to try and use different scanners with the cellphones, to enable them to scan data from tags pasted on objects. The scanned data will then be used for determining actions to be done in the application. Since scanning technology mostly has been used in industrial appliances for data collecting or handling, this will be a new way of using this technology, and we want to see if it will work out well for such an usage. We’re going to look at different types of scanning technologies, amongst others RFID and semacodes.

We hope to be able to plan and develop an application to test out the technology, and to let kids with Down syndrome test it out to see if it actually does help them in using a cellphone.

1.3 User scenarios

The following scenarios are examples of what we’re hoping to achieve with the application we’re proposing in this project. Eric is a fictional character who is 14 years old and has Down syndrome. The scenarios will describe imagined real life situations where he could use our proposed mobile application.

1.3.1 Scenario 1: Communication with parents from school

Eric is at school and finds out that the last two lessons are cancelled today, so he can go home earlier than usual. He picks up his notebook with drawings and text, and looks up the page with the drawing of him and his dad in their car. For him this drawing means that he want to go home. He then takes out his cellphone, and uses the scanner attached to the cellphone to scan the tag next to the drawing. The cellphone then shows a photo of his dad, with a text message saying “I can go home now.” Eric presses the “Ear” button on the screen to listen to the text read out loud by the phone application, to make sure that it is the correct message. Then he presses the green phone button to send the message. When his dad receives the message, he sends a message back to Eric. The message shows up on Eric’s phone, with a photo of his father. Eric then presses the “Ear” button to listen to the message, which says “I am coming soon!”
1.3.2 Scenario 2: Communication with friends from home

Eric has finished his dinner and wants to go play with his friends. He opens up his notebook and looks up the page with his drawing of his best friend, Thomas. He uses the scanner attached to his cellphone to scan the tag next to the drawing, and the phone shows a picture of a phone and a photo of Thomas. Eric then presses the green button, and the phone starts dialling Thomas’ phone number.

1.3.3 Scenario 3: Configuration of the application

Eric has gotten a new friend at school, and wants to be able to call him. He makes a drawing of his friend in his notebook. When he’s home he tells his father about his new friend, and shows him the picture he made. His father then pastes a new tag on that page, and borrows Eric phone to add his new friend to the application. His father then goes into configuration mode with a code, and adds the friend to the contact list. Then he scans the tag he put on the drawing, and sets the application to call the friend when that tag is scanned. Now Eric can call his friend!

1.4 Project context

The project will be done in collaboration with Steinar Brede at “Telenor Research and Development”. He will help providing the hardware and software for the technologies we’re using, and help with making design choices for the application.

The main goal for this project is:

- Create an application which helps kids with Down syndrome to use a cellphone.

To achieve this, we have these secondary goals:

G1. Do research on technologies which can be used for our purpose, and select one (or more) to use in our application
G2. Plan and develop the application
G3. Have users from our target group test the application, to see if it does help the kids:
  - Is the application intuitive?
  - Is it easier to use than a normal phone?
  - Does using this help them communicate more with their friends and family?
We aim to be able to fulfil all three secondary goals. For G1, we will have to look at the advantages and drawbacks to each technology, and we will also have to see if it is at all possible to implement the technologies in our application. With G2 we aim to be able to develop a stable and functional application which will be able to fulfil our main goal. G3 will determine if the main goal was actually reached. We will let users from our target group test the application, and then have them fill out a feedback form, in order to see if the points under G3 were fulfilled.

1.5 Research methodology

To be able to reach our goals, we need to find a research methodology that fits this project. Methodology has often been a problem in software development processes, and was addressed as a crisis by Glass [1]. He has summarizes four research methods:

**Scientific method**

*Observe the world, propose a model or theory of behaviour, measure and analyze, validate hypotheses of the model or theory, and if possible repeat.*

**Engineering method**

*Observe existing solutions, propose better solutions, build or develop, measure and analyze, repeat until no further improvements are possible.*

**Analytical method**

*Propose a formal theory or set of axioms, develop a theory, derive results, and if possible compare with empirical observations.*

**Empirical method**

*Propose a model, develop statistical or other methods, apply to case studies, measure and analyze, validate the model, repeat.*

The method that we are going to use in this project is best classified as an engineering method. As we already mentioned in the project context, the method for our project will be divided into following parts;

1. **Look at existing technologies and projects, and decide which of those technologies we can use for our project and looking into outcomes of other projects.**
Looking into projects done earlier and gathering input from other resources, we will evaluate technologies that are available to us, consider them and hopefully do a proper choice of usages for this project. There might be solutions already used in earlier projects that we could look into in order to propose better solutions. Finishing this part of the project will fulfil the secondary goal G1.

2. **Plan the architecture for a cellphone application using the technologies we suggested.**
   After researching and choosing the technologies, we will write a requirement specification and make the architecture for a prototype application. This application will be used to reach our main goal. This step will fulfil part of the secondary goal G2.

3. **Develop the application and test it to see if the goals were reached.**
   With the architecture at place we will develop the prototype and make it ready for the testing. The user testing will be observed and a report will be written. Finishing this part of the project will fulfil the second half of the secondary goals G2, and the first half of G3.

4. **Evaluation of the research.**
   When data is gathered from the user tests, we will analyse the outcome and write an evaluation based on our goals. Finally we will write a conclusion and propose suggestions on further work that could be a step towards a future project for making a more suitable product. This will conclude the work for the secondary goal G3, and if the results are positive, will fulfil both G3 and the main goal of the project.

**1.6 Outline of the Report**

This report will describe how we are going to plan this application, how we are developing it, and the results of the development. The report is divided into these parts:

- **Introduction:** This chapter, describes the background, motivation and how we’re planning to work with this project.
- **Prestudy:** Describes the technologies that we’ve considered in the project, and which one we’ve decided to use
- **Requirements:** Describes how we want the application we’re going to develop to work.
- **Architecture:** Describes how we’re planning to build the application
- **Development:** Describes the final version of the application that we’ve developed, and any problems we’ve encountered and how we solved them
• **Evaluation:** Evaluates our research method, and evaluates our application against our requirements and architecture. Also discusses if we’ve reached our goals.

• **Conclusion:** Discusses how well the project turned out.

• **Future work:** Describes how the project can be extended.

• **Glossary:** A list of words used in this report and their explanations.

• **References:** A list of the sources that we’ve used in our project.

• **Appendices:** Additional documents supplements that we’ve used in this project.

2 Prestudy

In this chapter we’ll describe the research that we’ve done in the start of the project. We have done research on our target group, and on the technologies we’ve considered to use. For the technologies we’ll write background information about each, why we’ve considered them, and the advantages and drawbacks for each of them. The last subchapter will be a summary of the choices we’ve made, and list up the technologies that we’ve chosen to use.

2.1 Related work

First we will introduce some other projects related to our project. The first project that we had a closer look at is the EiT project done by students at NTNU in spring 2005. Our project is to some degree a continuation of the EiT project, which therefore has some influence on the work in our project. Another project that we have studied is the plant scanner which was done as a thesis at Michigan State University. This project used a RFID scanner on a handheld device to interact with the application. This usage is quite similar to what we want to achieve, with the only difference being what we want to use it for.

2.1.1 EiT project [2]

This project was done by 23 students attending the Interdisciplinary teamwork (EiT) program at NTNU which is an obligatory course in the master’s degree course. The project group made a handheld mobile device application that would help children with Down syndrome. The application was divided into three submenus: a phone menu,
a calendar menu, and a message menu. From the phone menu the child could choose a person from a list of pictures and initial a phone call to the selected one. The calendar menu would present a vertical list of activities made of picture symbols, and mark the one that is happening at the current time. The child could also look at the future activities. The last submenu was the message menu where the child could select a message from a list of picture symbols and send it to one of his/her contacts. Each picture symbol represented a message, for example a picture of a car would have a message “I want to come home”.

To be able to configure all information used in the application, they chose to make a portal where the parents could configure the application. Whenever a parent wanted to add a contact, make a new picture symbol or change/add text messages in the message menu, they would have to do it from the portal, and the changes would then be sent to the device using GPRS.

**Result of the project**

The handheld mobile device was not fully implemented and did not cover all of the requirements that were given. However, the main requirements were fulfilled and a working application was implemented. When reading the document we could not find any chapter that dealt with architecture or any architectural process, which could explain the difficulties they encountered. There were also some functionality that weren’t completed because of the limited time they had available, and the difficulties they had getting information about mobile device programming.

A final user test was not executed due to the time limitations and thus there were no statistics that would tell us if this project worked or not. It seems that the main problem in this project were the time limitations. Leaving us to no specific conclusion, we had to analyse the project and make our own conclusion, and try to avoid the problems they seem to have encountered when we’re planning our own project. From studying this report and asking some of the project members, we’ve come to two conclusions:

- Make sure to have a good architecture before development.
- Have a user interface which doesn’t rely on lots of built-in symbols for interacting
2.1.2 Plant scanner [3]

The reason that we wanted to take a closer look at this project was its usage of a RFID scanner with a handheld device, and also that it targeted children. The Plant scanner project uses a handheld device (PDA) to teach children introductory plant biology and to enhance their exploration of the indoor children’s garden.

One example: A child walks through the garden with a handheld device and stops in front of a plant – let’s say a tomato plant. In front of the tomato plant is a physical sign. If the child waves the PDA over the tomato sign, information about the tomato plant appears in the interface.

The sign next to the plant has an RFID tag, and the handheld PDA has an RFID reader, so when the child waves the PDA in front of the sign, data is passed to the PDA.

An implementation of the plant scanner was done and a user testing was performed. The children got their own PDA and went on trying it, and were later interviewed about their experience with the device. The results indicated that the plant scanner was a success. The children had no difficulties holding the plant scanner and using it. Even the eight year old did not have difficulties using the plant scanner. The presence of an involved guide combined with the plant scanner heightens child interest, involvement, and learning from the experience.

While this project was done with children without Down syndrome, we are hoping to get to the same result and conclusion in our project. Even though the usage of the technology in the plant scanner project is quite similar to ours, we believe that we are under much stricter constraints in our project. Nevertheless, we will see if our results are as successful when our project is finished and a conclusion is written.
2.2 Down syndrome

2.2.1 Research from written material

Before doing anything else in this project, we felt that we needed to know a bit more about our targeted group. Since we are going to make architectural and design decisions later in this project, we found it necessary to know more about the domain. This is a very important stage of a successful software development project. What we have done is to collect some articles about Down syndrome that would be helpful to our project. These articles contain research on ocular disorders and empirical research to investigate the literacy, language and memory skills of children with Down syndrome. By reading these articles we hope to get a better picture of what we have to take in account when making architectural and design decisions, and hopefully not underestimate or overestimate the needs of our target group, which could lead to an unusable application.

Reading article [4] we discovered research facts that would help our understanding of the domain. The article used an empirical method to compare 24 children with Down syndrome against two groups of children selected from their classmates. The comparison groups were a group of typically developing children who were average readers in the classes and a group of children who were matched to the children with Down syndrome for reading age. The groups where measured in reading, spelling, short term memory and numbers. In short, the results indicated that the children with Down syndrome fell significantly behind the other groups on measures of language, memory and number skill. However, the children with Down syndrome appear to have advanced reading ability compared to their other cognitive abilities.

The other research paper we read showed that the short-term memory span is usually not increased with age at the typical rate, and most teenagers and adults only have spans of two to four digits. [5]

Having studied these articles, we have realised that we have to take these cognitive abilities into account when going further into the project. We will try to make the design as simple as possible and not forcing the users to process a vast number of screens in order to do an action. This is to make the application easier and more usable for our target group.
In another article [6], 68 patients with Down syndrome were enrolled in a study pertaining to sight disorders. The information was obtained from previous sight examinations and parents completing a questionnaire. The results indicated that out of 68 patients, 12 had poor vision on both eyes and 15 patients were found to be amblyopic. This study suggests that children with Down syndrome may be at a greater risk for visual impairment. It was also stated that if the child is mentally retarded, as most individuals with Down syndrome are, an additional handicap or sensory impairment may further limit the child’s overall functioning and may prevent the child from participating in significant learning process.

After reading this paper we decided that there should be some certain criterias that we’d like for the cellphone we’re going to use. One of them would be a big screen so that the graphics can be as big and noticeable as possible. If the screen is also a touch screen, it’d also be possible to have large graphical buttons on the screen. This is important since most of the commercial cellphones either have small screens or small buttons.

### 2.2.2 Information from parents

The empirical research articles are good sources of information about Down syndrome, but we also have our teaching supervisor as an information source in this domain. Steinar Brede, our supervisor, is himself a parent of a girl with Down syndrome, and they are members of the Ups&Downs Sør-Trøndelag association, an association for families with kids having Down syndrome. Steinar Brede was therefore of great help when we were doing the design decisions for our application.

### 2.3 Context

To make sure that our application will work in different situations and settings, we need to know the context of our target group. However, to know the context we need to have a specific definition of context. There has been given many definitions by different authors. Brown et al.[7] define context as location, identities of the people around the user, the time of day, season, temperature, etc. Ryan et al.[8] define context as the user’s location, environment, identity and time.

Dey [9] specifies context as the user’s emotional state, focus of attention, location and orientation, date and time, objects, and people in the user’s environment. Schilit et al. [10] claims that the important aspects of context are: where you are, who you are with, and what
resources are nearby. They define context to be the constantly changing execution environment.

They include the following pieces of the environment:

- **Computing environment** available processors, devices accessible for user input and display, network capacity, connectivity, and costs of computing
- **User environment** location, collection of nearby people, and social situation
- **Physical environment** lighting and noise level

Anind K. Dey and Gregory D Abowd [11] have described context as:

*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 applications themselves.*

We find that the last definition by Dey and Abowd to suite us best and will continue to elaborate on that. The main entity in this project is the children with Down syndrome. The situation about our entity is rather static since children with Down syndrome are dependent on a schedule planned by their guardians. The information that can help us characterize the situation of an entity will be the guardians who is planning and maintaining the schedule.

Information of this kind is essential to design decision on making the application proper interact with the user.

### 2.4 Mobile phones

Since we have to consider all of the issues mentioned in the previous subchapters, and we also need to use a scanning technology with our application, our choice of cellphones was limited. The two choices we ended up with were Qtek S100 and Nokia 5140.

#### 2.4.1 Qtek s100

This is a PDA phone manufactured in beginning of year 2005. The size of this PDA is in the same class as Nokia Symbian models and P900-serie from Sony Ericsson. The Qtek has a powerful CPU and large memory compared to normal cellphones, and that is also the
reason that it’s somewhat larger than normal cellphones. It has a built in camera and Bluetooth. The specification for this model can be found in the appendix.

The Qtek doesn’t have a built-in scanner, so we’ll have to use some kind of add-on device to be able to do any scanning with it. However, it does have the big touch screen that was suggested in the section about Down syndrome, while it isn’t as big or expensive as some other Qtek models, which is the reason this phone was one of the choices.

The Qtek uses the Windows CE operating system, and development for the Qtek will be done with the .NET Compact Framework, although the Qtek also supports J2ME (Java 2 Micro Edition).

2.4.2 Nokia 5140

This model from Nokia has a RFID reader with the NFC technology, installed on the back of the cellphone. This phone is therefore more practical than the Qtek, because we don’t have to use an add-on device for doing scanning, although this locks us to using RFID as the scanning technology for our application. Another drawback is that the screen is significantly smaller, and it has a normal keypad instead of a touch screen.

The Nokia 5140 uses the Nokia Series 40 platform, and development for Nokia 5140 will be done using J2ME.

2.5 Programming languages and development environments

There are two main development environments that we can choose between, which we’ve already mentioned. The first one is the .NET Compact Framework, which is Microsoft’s programming environment for Windows operating systems. The other one is Sun Microsystems’ J2ME, which is supported by most modern cellphones. We’ve also written about the operating system for each cellphone.

2.5.1 Windows CE

The Windows CE operating system is based on the Win32 application programming interface (API). The fundamentals of programming for Windows CE closely parallel programming for
Windows 98, Windows NT, and Windows 2000. As with the other Windows operating systems, Windows CE is an event-driven programming model. A Windows CE-based program receives messages, interprets the messages, and acts on the messages.

### 2.5.2 Visual C++ for Windows CE

Visual C++ for Windows CE is simply an add-on to Visual C++. For those who have used Visual C++ to create Win32 applications or components (using Active Template Library (ATL), Microsoft Foundation Classes (MFC), or the Win32 API), creating applications or components for Windows CE will not be that much of a difference.

### 2.5.3 Compact Framework .NET [12]

The .NET Compact Framework has been developed from mainly three trends the last decade. The first trend was the popularization of sophisticated development and runtime environments that increase developer productivity and application robustness. These environments offer features such as integrated type safety at runtime, memory management, and a standardized, rich programming framework. These environments got even more powerful when combined with development tools specifically designed to host and target their runtime environments. Microsoft Visual Studio .NET and the .NET Framework are such a combination. The .NET Compact Framework was the natural extension, bringing this high-productivity environment to devices with more limited resources than desktop PCs.

A new trend for developers is that they have started writing applications targeting small devices, in order to provide information and services in a wider variety of situation. One major barrier for developing applications on these types of devices has been the deficiency of developer’s skills and knowledge. Both .NET Compact Framework and .NET Framework has been based on the ECMA Common Language Infrastructure standards. This means that both source code and compiled binaries can run across Windows based desktops and servers and various other devices and processors, thereby offering improved productivity and efficiency. This is not to say that applications written for a server would necessary be useful running directly on a device.

The last trend was the increase in SOA (Service Orientated Architecture) using standardized protocols like SOAP and WSDL (Web Service Desription Language). The .NET Compact
Framework was designed from ground up so that it could easily access remote data using these Internet standard protocols.

In contrast to the .NET Framework, which is mainly for the desktop or server versions of Windows, the .NET Compact Framework is designed to run on a variety of different operating systems and has a unique architecture to provide for this cross-OS functionality. Choosing this framework, we have to program in C# or Visual Basic .NET. Of those two languages, C# is the one most similar to Java, which we are more familiar with. C# is therefore the most likely choice for our project.

2.5.4 Series 40 (Nokia)

Nokia Series 40 Platform is a is a set of technologies and APIs that is common for all Series 40 cellphones, making development for these phones easier. Some of the included things are Java APIs, Multimedia Messaging Service (MMS), and audio and video support. Not everything is similar though, the screen resolution and keypad on each phone may differ, and some phones may have features that others don’t have.

Series 40 cellphones doesn’t have a “native” programming environment, so all development for Series 40 cellphones are done with J2ME.

2.5.5 Java 2 Micro Edition (J2ME)

J2ME is a version of Java for handheld devices. Java is a software technology developed to provide a common base for running applications across a wide variety of computing platforms. It consists of a programming language and a run-time environment.

Java applications can run on servers, personal computers, mobile phones and other devices. Because the capabilities of these computing platforms vary, Java has evolved into 3 editions:

1. Java 2 Enterprise Edition (J2EE) - for high-end business machines
2. Java 2 Standard Edition (J2SE) - for personal computers
3. Java 2 Micro Edition (J2ME) - for smaller handheld devices such as mobile phones and PDAs

The Mobile Information Device Profile (MIDP) is a key element of the J2ME, defining a platform for dynamic networked applications. Developers using MIDP can write applications
once, and then deploy them to a wide variety of mobile devices. MIDP has been widely adopted as the platform of choice for mobile applications.

While J2ME is designed to be portable, it doesn’t have as good access to core functionalities (making phone calls, sending SMS) as the .NET Compact Framework has for the Qtek phone. We will therefore not use J2ME when developing for Qtek. If we choose to develop the software for Nokia 5140, however, we won’t have other alternatives than to use J2ME, as we mentioned earlier in the Nokia 5140 subchapter.

2.6 Technologies for scanning data tags

In order to reach the goals that we want to address in this project, we’ve been looking for different technologies that allow scanning of different types of data tags. The reason we want to use data tags, is that they can practically be attached to anything. This makes it possible to “identify” simple things like kids’ drawings or household objects in a database, and link them to actions or messages. Since it’s difficult for kids with Down syndrome to use a normal phone, we want to try to use this method for user input for our application, to see if it’s an improvement in usability.

In this part of the document we’ll write a bit about each scanning technology we’ve considered, and the advantages and drawbacks for each of them.

2.6.1 RFID

RFID (Radio Frequency IDentification) is a method of storing and remotely retrieving data using devices called RFID tags. A RFID tag is a small computer chip that may be used for identification when attached or incorporated into a product, animal, or person. RFID tags contain antennas which enables them to receive and respond to radio-frequency queries from an RFID transceiver. There are two different kinds of tags: passive and active.

Passive RFID tags have no internal power supply. When an incoming radio frequency signal is provided by the transceiver, an electrical current is induced in the antenna, and this current provides just enough power for the tag to transmit a response. Due to the limited power and cost, the response of a passive tag is not much more than an identification number. The lack of onboard power supply means that the tags can be quite small: the smallest such devices commercially available measured 0.4 mm × 0.4 mm. These tags have practical read distances ranging from about 10 mm to about 6 metres.
**Active RFID tags** have an internal power source, and may have longer range and larger memories than passive tags. This on the other hand makes them bigger in size. At present the smallest active tags are about the size of a coin. These tags have a practical read distance range of tens of metres, and a battery life of up to 10 years.

Because passive tags are cheaper to manufacture and have no battery, the majority of RFID tags are of the passive variety. While the cost advantages of passive tags over active are significant, other factors including accuracy, performance in certain environments such as around water or metal, and reliability make the use of active tags very common today.

### 2.6.1.1 IDBlue

The IDBlue is a RFID scanner shaped as a (thick) pen, and it connects to the cellphone using Bluetooth. This device was available at Telenor, so we were able to use it in our project. Working with this device, we could easily work with any cellphones that have Bluetooth support. However, since we now are concentrating on the Qtek s100 and Nokia 5140 cellphones, we will use this device with the Qtek s100. As mentioned before, Nokia 5140 has a built-in RFID reader so this device is not necessary.

Using this device in this project would not be optimal solution, since it is a standalone device, meaning that the users would need to handle two separate devices, a cellphone and a RFID reader pen.

### 2.6.1.2 SD card RFID reader

While writing this document, TradeWind Technologies introduced the world’s first RFID reader in the SD card form factor. One can now use PalmOS or PocketPC handheld device as an RFID reader to read or write many standard and proprietary 13.56Mhz tags and labels. The reader fits into any standard SD (SDIO) slot and extends outside the slot for a small external footprint of 28mm x 28 mm. This device is available for $1499 including the SDK. At this price we had no choice other than to leave out this reader from our project, and hope for better price or alternative in the future.
2.6.2 Semacode [13]

Semacode is described as a system for “everywhere” computing, that is to bring the computer out of the box and into the environment. Semacode is not a hardware solution, but a software solution that provides the tools necessary to build applications that combine aspects of the virtual world into the real world. It works by combining existing standardized elements such as camera phones, optical barcodes and URLs into an integrated system. Technically, a semacode is an optical barcode that contains a URL internet address. Using the Semacode Reader technology, a user can scan a semacode tag, and then with a single click connect to the web page on their portable camera phone.

Semacode tags such as the one pictured here are based on the international ISO/IEC standard Data Matrix. A semacode tag uses this type of optical 2D barcode to store a URL in a secure format that can resist up to 50% damage to the tag. Applications developers and general users can create their own semacode tags using the Tagger tools available from their website.

![Semacode Tag](image)

*Figure 6: Example on a semacode tag*

The way a semacode works is that the user has to take a picture of the tag, using an application which has implemented the semacode software. The application then decodes the picture and extracts the URL (or any text which is encoded in the semacode, even if it’s not a URL). The semacode software is supported by most Java cellphones, and there is also a version for Symbian (a cellphone OS used by Nokia and some other companies).

Most new cellphones have cameras, meaning there is no need to buy an additional scanner device. In addition, anyone with a computer, a printer and some computer knowledge can print their own semacode labels. This technology is therefore most certainly the cheapest one to implement, out of the ones we’ve suggested. However, a test of this technology showed
that it did not work as well as we had in mind. The picture taken by the phone camera had to be quite accurate for it to work, and the software took 4-5 seconds to extract the text from the picture tag after each scan. What we need in this project is a scanning technology that is quick and easy to use. This is something Semacode did not quite achieve, and semacode is therefore not suitable for this project.

### 2.6.3 Barcode reader

The Secure Digital Scan Card (SDSC) is a scanner that plugs into a Pocket PC’s slot for SD (Secure Digital) memory cards. It uses a Linear CMOS Imaging technology and supports a number of different barcode standards. Since our Qtek s100 phone has a SD slot, this reader is an option for our project, although it’d only work with the Qtek.

While not as cheap as the semacode alternative, this option is still cheaper than RFID, since like with semacode it’s quite simple for people to make their own barcode labels. It is therefore interesting to see how well scanning works with this reader.

A test of the barcode scanner revealed that as long as we used some specific barcode standards, the scanner would scan the barcode quite quickly. There is however no SDK or API available for the barcode scanner, since it is designed to send read data to any applications as virtual keystrokes. That means that the data read will be input to any text input field in any application. This shouldn’t be that much of a problem, so we’ll definitely try this scanner for our application.

### 2.7 Technologies for functionality

#### 2.7.1 Phone calls, SMS and MMS

The ability to make phone calls is one of the more basic functions of a cellphone. While not everyone in our target group may need this functionality, because of disabilities to express themselves with speech, it should still be an option.
**SMS (Short message service)** is a service available on most digital mobile phones that permits the sending of short messages (also known as SMSes, text messages, messages, or more colloquially texts or even txts) between mobile phones, other handheld devices and even landline telephones. Text messages have become quite popular since it was commercialized in 1992. For our application, we want to give the parents an option to set predefined SMS messages that the kids can send.

**MMS (Multimedia Messaging Service)** is a technology to send more advanced messages between cellphones via GPRS. People can add images and audio clips to the messages, and it can also have more text than the 160 characters limit of SMS. An MMS is instead limited in the total size of text, images and audio that is added to the message. By allowing MMS, we can just let the parents send images or audio clips to the kids instead of text, avoiding the need for having the software translate text to symbols. Almost all new cellphones have the option to send MMS, and to record audio clips to be sent.

In Visual C++ for Windows CE there is API support for handling phone calls and SMS messages, while there isn’t much direct support for this in the .NET Compact Framework. Implementing those functions in Qtex will therefore probably be done by first creating the libraries for the functions in C++, and then import these libraries in our C# application. Some libraries from OpenNETCF.org [14] can also be used for some of the phone functions.

With J2ME and using Wireless Messaging API we should be able to send and receive SMS messages. With MIDP 2.0 it is possible to make phone calls by invoking a special method, and the Series 40 SDK v2 supports MMS processing.

### 2.7.2 Text-to-Speech

Text-to-Speech is a technology/technique to take a written text and generate an audio version of it. There are two main solutions used for this: one is to put together prerecorded voice samples after a set of rules. The other is to generate audio using an acoustic model. The result is an audio clip which sounds like a spoken version of the text, in a specific language. The first solution usually gives a more natural sounding voice, but the speech engine takes more storage space because it needs to store all the audio clips used. The second solution gives a more robotic sounding result, but the speech engine can be smaller. This solution is
commonly used by people with visual impairment, because this solution can generate quicker/shorter audio clips which are still possible to understand.

In our project we’re considering text-to-speech as an option for converting messages for kids with Down syndrome. This way kids who can’t read SMS messages sent to their phone, can still understand them. Text-to-Speech can also be used to read the names of the people calling to the kids, so they can know who is calling them.

The drawback of using Text-to-Speech would be that it will require quite a lot of resources and memory, so it’s most likely impossible to do on Nokia 5140. An unnatural voice may also alienate the kids, although we’re not certain how much of a problem this would be, partly also because some voice engines have gotten quite good in our days.

To add Text-to-Speech in our application, we’ll try and obtain a 3rd party engine, which can be imported into our application.

2.7.3 Push-to-talk

Push-to-talk is a technology to allow people to use the cellphone as a walkie-talkie: they can set up a group of people/cellphones, and when they use the push-to-talk button, they’ll be talking to everyone in that group. Only one person can talk at a time.

According to an article on the Java Community Process website, [15], it seems that it is currently not possible to implement Push-to-talk in Java. We haven’t found much information about implementing this technology in C# either, so it will probably not be implemented.

2.7.4 Text-to-video

Since we know that many kids in our target group have problems reading text, we have suggested using Text-to-Speech to convert text to audio. But since some of the kids also are hearing impaired or for other reasons have difficulty understanding speech, we want to see if we can come up with a solution that would help them too.

A suggestion that has been brought up is to have a library of short video clips where words are shown in sign language. With this library, we could “translate” a received SMS message into a video showing the words in the message with sign language. This solution would help
those who have difficulty understanding speech; however, we’re not certain that video played on a small cellphone screen would be easily understandable. Also there’s the issue that video playback does require some processing power and storing the videos also require lots of storage space, meaning it’s most likely not possible on the Nokia 5140. After doing a bit more research on this, it also turns out that video playback isn’t directly supported in the .NET Compact Framework, and that implementing it would require quite some work. So unless we get time left after implementing other functionality, we most likely won’t have enough time to try implementing this.

2.7.5 Text-to-symbols

Another method similar to text-to-video would be a method where we use a library of symbols instead of videos. There would have to be a set of rules describing which words or groups of words translates into what symbols.

Since the people in the EiT project found that symbols made by “strangers” were hard to relate to for the kids, this solution wouldn’t be very good, so we’ll probably not use this solution.

2.7.6 Camera

Many newer cellphone models have a built-in camera. While the quality on the cellphone cameras usually isn’t quite as good as normal digital cameras, it’s still usable for getting photos on the phones. We want it to be possible to set up our software so that it shows a photo of whoever is calling or sending the kids a SMS. For that we will use the camera. The parents should be able to take a photo for each contact in the contact list, so the kids can distinguish between them by looking at the photo.

If we choose to add MMS support in our application, the camera could also be used to allow the kids to take photos and send them as messages.
2.8 Data storage

2.8.1 XML

eXtensible Markup Language (XML) is a tag language designed to be portable between applications. XML files are basically clear text files, where the information is put inside tags. An extract from an XML file could look like this:

```xml
<contact>
  <name>Some Person</name>
  <address>Some Where 123</address>
</contact>
```

The text with <>-brackets are called tags, and are used to categorize the information.

We will use XML to store information used in our application, because XML is a widely accepted standard for storing information, and it will be easier to copy the information from our software for use in other software. There is also built-in support for XML in C#, which in itself is a good reason to use it.

2.8.2 Client-Server

Client-server is a way of splitting up an application in order to divide the workload on several units, and to allow the same information to be changed and shared between several units. The server usually does “most of the work” and is located on a powerful computer, while the client is located on the users’ computers/devices, and fetches information from the server for showing to the user.

For our application, it is possible to create an external server for storing information, instead of storing it locally. The application would have to connect to the server when it needed information. The advantage of this is that the information could if necessary be updated without having access to the cellphone that the kids are using. Any heavy computing could also be done on the server, and the result could be sent to the phone application after computing. The drawback is that it’ll be a delay every time information is needed, because the phone has to connect to the server and wait for the necessary information to be sent back. This delay could also possibly eliminate the advantage of doing the computing on a more powerful computer.
2.9 Choices

We’ve chosen to develop our application for Qtek s100, and leave out the Nokia 5140. The reason is that there are too many limitations in the Nokia phone for it to be practical in our project. We are going to use Microsoft Visual Studio .NET, and develop our application using C#, with some libraries coded with eMbedded Visual C++ where necessary.

These are the technologies we’ve decided to add in our application:

- Barcode scanning
- RFID scanning
- Making and receiving phone calls
- Sending and receiving SMS
- Text to speech
- Camera
- XML

We’re going to use a barcode scanner and RFID scanner for reading tags. The reason we didn’t choose semacode, is that it is most likely too slow and difficult for kids to use. The barcode scanner and RFID scanner are quicker and more suitable.

The functionality of making and receiving phone call are rather necessary core functionality for a phone, and SMS is also so common that it is necessary. MMS would’ve been useful too, but unfortunately we couldn’t find any information on implementing it. There weren’t much info about implementing Push to talk either, and even if there was, we aren’t sure it would’ve been of much use for our target group. Therefore we have decided not to implement MMS and Push-to-talk in our application.

Text-to-speech will mainly be used for reading SMS messages, for kids who can’t or have difficulty reading. We’re going to use a Text-to-Speech evaluation engine from the Acapela Group [16]. Text-to-video was too demanding for our application, and may have required a more extensive server-client architecture in order to patch the videos together. Otherwise there would be a noticeable lag between video clips, making it difficult to understand the connection between the clips. Text-to-symbols were also left out because it would require a large library of words and the symbols they are connected to, and we wouldn’t be certain if
the kids would be able to relate to symbols that “strangers” have made. Besides, part of the reason we wanted to use tag scanning, was to have as little information on the mobile screen as possible.

The camera will be used for adding photos of contacts, so the kids can see who is contacting them or who they are contacting. XML will be used to store all the data used in the application. We don’t want to use a server-client architecture, because using a server for storing information would give too much of a lag in the application when it needs to fetch information. We also haven’t chosen any technologies which require too large amount of data processing, so it would not be any advantages to having a server to store the data.

3 Requirement specification

This part of the document describes the requirements we have determined for our application. The requirements were decided based on our prestudy and our dialogue with our supervisor Steinar Brede.

3.1 Functional requirements

These are the requirements that define how the application should work, and how we expect it to be used by the kids and the parents. Note that some of the requirements have several alternatives, where it is only necessary to choose one of them. Although it may be possible to implement more than one of the alternatives, we will probably only choose one of them, because of time limitations.
3.1.1 Use case diagram for users (children)

3.1.2 Requirements for users (children)

1. The application should be able to store tag data and connect those to specific actions:
   1.1. Send a predefined SMS, either:
      1.1.1. to specific person(s)
            or:
      1.1.2. choose from address book (with pictures)
            or:
      1.1.3. to a person identified by another tag
   1.2. Make a phone call:
      1.2.1. to specific person(s)
            or:
      1.2.2. choose from address book (with pictures)
            or:
      1.2.3. alternative: if a person’s tag is scanned without scanning a SMS tag first
2. The application should also be able to receive communication:
   2.1. Receive phone calls
   2.2. Receive SMS
      2.2.1. Should be deleted after it has been read.
   2.3. Receive MMS
3. The application should have an address book with pictures in order to allow the following functionality:
3.1. Should identify the person who is calling with more than just text:
   3.1.1. by showing a picture
       and/or:
   3.1.2. by reading the name (text-to-speech or stored audio clip)

3.1.3 Use case diagram for the configurators (guardians)

3.1.4 Requirements for configurators (parents)

4. The application needs a configuration mode where parents can:
   4.1. Add new tag data, and connect these to specific actions (specified in requirement 1.)
   4.2. Modify existing data and connections
   4.3. Delete tag data
   4.4. Add a person to the address book, by taking a photo of him/her (or a drawing) with
       the phone camera, and then adding details:
       4.4.1. name
       4.4.2. number
       4.4.3. voice clip (person’s name)
   4.5. Edit a person’s information, change the same details as in the previous point, and
       change picture.
   4.6. Delete a person
   4.7. Should be able to lock the phone to only receive calls from specific numbers
## 3.2 Quality Requirements

### 3.2.1 Usability

**Scenario A:** Deciding between options

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>External</td>
</tr>
<tr>
<td>Stimulus</td>
<td>The kid wants to do something, and needs to decide which menu option to choose</td>
</tr>
<tr>
<td>Artifact</td>
<td>GUI</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation</td>
</tr>
<tr>
<td>Response</td>
<td>The time used to find the correct option</td>
</tr>
<tr>
<td>Response measure</td>
<td>The kid should be able to find the correct option within 10 seconds, and a maximum of 2 menu screens to get to it. (Optimally only 1.)</td>
</tr>
</tbody>
</table>

**Scenario B:** Activate correct menu options

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>External</td>
</tr>
<tr>
<td>Stimulus</td>
<td>The kid tries to press one of the menu options in the GUI</td>
</tr>
<tr>
<td>Artifact</td>
<td>Touch screen</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation</td>
</tr>
<tr>
<td>Response</td>
<td>Should activate the correct menu option</td>
</tr>
<tr>
<td>Response measure</td>
<td>Error margin should be 0%</td>
</tr>
</tbody>
</table>

**Scenario C:** Adding new and editing existing RFID tags and actions

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>External</td>
</tr>
<tr>
<td>Stimulus</td>
<td>The parent adds a new data tag and action, or edits an existing tag and action</td>
</tr>
<tr>
<td>Artifact</td>
<td>GUI</td>
</tr>
<tr>
<td>Environment</td>
<td>Configuration mode</td>
</tr>
<tr>
<td>Response</td>
<td>A new data tag and action is stored, or the new action for a tag is stored</td>
</tr>
<tr>
<td>Response measure</td>
<td>The whole configuration (minus the time it takes to write eventual SMS messages) should be possible to do within 1 minute.</td>
</tr>
</tbody>
</table>
### 3.2.2 Modifiability

**Scenario D:** Use different phone models

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Internal</td>
</tr>
<tr>
<td>Stimulus</td>
<td>We’re using a different phone model</td>
</tr>
<tr>
<td>Artifact</td>
<td>System</td>
</tr>
<tr>
<td>Environment</td>
<td>Development</td>
</tr>
<tr>
<td>Response</td>
<td>The screen resolution should be accounted for</td>
</tr>
<tr>
<td>Response measure</td>
<td>No parts of the GUI should be cut out, and there shouldn’t be large blank unused spaces.</td>
</tr>
</tbody>
</table>

**Scenario E:** Use different scanners

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Internal</td>
</tr>
<tr>
<td>Stimulus</td>
<td>We’re using a different scanners</td>
</tr>
<tr>
<td>Artifact</td>
<td>System</td>
</tr>
<tr>
<td>Environment</td>
<td>Development</td>
</tr>
<tr>
<td>Response</td>
<td>Should be able to change only the scanner part of the code</td>
</tr>
<tr>
<td>Response measure</td>
<td>No other parts of the code should need to be changed.</td>
</tr>
</tbody>
</table>

**Scenario F:** Independability between parts of code

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Internal</td>
</tr>
<tr>
<td>Stimulus</td>
<td>We want to change one part of the code</td>
</tr>
<tr>
<td>Artifact</td>
<td>System</td>
</tr>
<tr>
<td>Environment</td>
<td>Development</td>
</tr>
<tr>
<td>Response</td>
<td>Shouldn’t affect other parts of the code</td>
</tr>
<tr>
<td>Response measure</td>
<td>Less than 10% of other code should be affected.</td>
</tr>
</tbody>
</table>

### 3.2.3 Availability

**Scenario G:** Uptime

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>External</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Normal usage and configuration mode</td>
</tr>
<tr>
<td>Artifact</td>
<td>GUI</td>
</tr>
</tbody>
</table>
Environment: Normal operation
Response: Program shouldn’t crash
Response measure: 100% uptime

**Scenario H:** Application should not be exited by the children

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>External</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Normal usage by the kid</td>
</tr>
<tr>
<td>Artifact</td>
<td>GUI</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation</td>
</tr>
<tr>
<td>Response</td>
<td>The children shouldn’t be able to exit program by themselves accidentally.</td>
</tr>
<tr>
<td>Response measure</td>
<td>0% chance of exiting.</td>
</tr>
</tbody>
</table>

**3.2.4 Testability**

**Scenario I:** Should be able to determine error location

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>External</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Error</td>
</tr>
<tr>
<td>Artifact</td>
<td>System</td>
</tr>
<tr>
<td>Environment</td>
<td>Testing</td>
</tr>
<tr>
<td>Response</td>
<td>Should be able to determine error’s location in code</td>
</tr>
<tr>
<td>Response measure</td>
<td>Within 5 minutes.</td>
</tr>
</tbody>
</table>

**3.2.5 Performance**

**Scenario J:** Immediately show result of menu option

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>External</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Choosing a menu option</td>
</tr>
<tr>
<td>Artifact</td>
<td>GUI</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation and configuration mode</td>
</tr>
<tr>
<td>Response</td>
<td>Should show the result of that menu option as soon as possible, or at least show a “loading” message.</td>
</tr>
<tr>
<td>Response measure</td>
<td>Within 1 seconds.</td>
</tr>
</tbody>
</table>
3.2.6 Security

**Scenario K:** Kids shouldn’t be able to enter configuration

<table>
<thead>
<tr>
<th>Part of scenario</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Extern</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Normal usage by kid</td>
</tr>
<tr>
<td>Artifact</td>
<td>GUI</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation</td>
</tr>
<tr>
<td>Response</td>
<td>Shouldn’t be possible to enter configuration mode by accident</td>
</tr>
<tr>
<td>Response measure</td>
<td>Less than 0.1% chance.</td>
</tr>
</tbody>
</table>

## 4 Architecture

This part of the document shows the architecture of the software. Any changes we’ve done in the architecture while developing the application has been changed in this part of the document too, in order for it to correctly reflect the most current version of our application when writing this document. We have used the book [17] as an information source for making the architecture.

### 4.1 Architectural drivers

Our most important architectural drivers will be **usability** and **modifiability**.

The reason we chose usability is because of the target group, which are kids with Down syndrome. The reason for modifiability is that we don’t want the software to be locked to a specific hardware solution. It should be possible to use it with another cellphone or scanner without having to change the whole program.

### 4.2 Stakeholders and concerns

This part of the document describes the ones who have interests in the application developed in this project, and what their concerns are.

#### 4.2.1 Users and acquirers of the system

The users of the system are the kids, who’ll be using the system for keeping contact with their parents and other guardians. The parents are both users and acquirers, as they’ll have to set up the phone for the kids to use.
These stakeholders have these concerns for the system:

1a) Should be useful/helpful in their daily life
1b) System should be easy to use
1c) Should be predictable
1d) Risk: The system could make things more difficult rather than easier.

4.2.2 Developers of the system

The developers of the system will be the project group for this project. The concerns are:

2a) This system is made to test out and make good use of new technology
2b) RFID and barcode scanners are still not in wide use amongst consumers, so even if the technology itself isn’t brand new, this usage of it is.
2c) Development of the system should be possible to split into parts, which can be done as independently as possible.
2d) Risk: Relatively new technology (The RFID and barcode scanners hasn’t been used much with cellphones) may make system unstable and unreliable.
2e) Risk: System may not meet target group’s need
2f) Should be easily understood by the users.
2g) Should work with the hardware that they’re using.

4.2.3 Maintainers of the system

The maintainers of the system are any person or teams who’ll be doing further development and/or maintenance of this system after this project is over.

3a) System should be usable even if the hardware is changed over time.
3b) The system should be possible to develop in such a way that it can be modified for use in other hardware, so it’s feasible.
3c) It should be easy to find out which part of the code does what in the system.
3d) Risk: Hardware changes more than the developers have taken into account.
3e) Risk: The system isn’t well enough documented.

4.3 Selection of architectural viewpoints

This part of the architecture shows what viewpoints we’re going to have in the architecture, and which stakeholders the viewpoints are for. We’ve chosen the viewpoints that are most relevant to us from the 4+1 View Model of Architecture [18].
4.3.1 Logical viewpoint

The stakeholders addressed by this viewpoint:

- Developers, maintainers

Concerns addressed by this viewpoint:

- 2c), 3b), 3c), 3e)

Modeling technique used for modelling this viewpoint:

- UML Class Diagrams

4.3.2 Process view

The stakeholders addressed by this viewpoint:

- Developers, maintainers

Concerns addressed by this viewpoint:

- 1c), 2c), 3b), 3c), 3e)

Modeling technique used for modelling this viewpoint:

- UML Activity Diagrams

4.3.3 Development view

The stakeholders addressed by this viewpoint:

- Developers, maintainers

Concerns addressed by this viewpoint:

- 2c), 3a)

Modeling technique used for modelling this viewpoint:

- Layered diagram

4.4 Quality tactics

This chapter describes the tactics we’ll apply while developing the application in order to achieve the quality requirements/goals that we have for the application.

4.4.1 Usability

1-A The GUI should be divided into two parts, which should be:

a. Main GUI: This GUI is the one used by the kids

b. Configuration GUI: This GUI is used by parents to set up the software, and shouldn’t be accessible for the kids.

1-B Main GUI should be easily understandable for kids:
a. Use large graphical buttons/menu choices
b. Don’t depend on text to describe menu choices
c. It should only be two menu choices available at any time
d. There shouldn’t need too many steps to do specific actions. Minimal input!

4.4.2 Modifiability

2-A Split code into modules for different parts of the functionality:
   a. Modules should be as independent as possible
   b. Dataflow between modules should be as small as possible

2-B Good code layout and comments:
   a. Indenting
   b. Comments before every function
   c. Comments at beginning of module, describing external functions and their usage/parameters
   d. Sensible variable names, avoid one-letter variables (to make it easier to search for variables)

2-C Follow standards:
   a. Configuration of the software should be stored in valid XML

4.4.3 Testability

3-A Create the GUI as early as possible (doesn’t need to be the whole GUI), so it is easier to test the functionality

4.4.4 Security

4-A Use a special key combination and/or use a security code to enter configuration mode.

4.4.5 Performance

5-A Avoid loading lots of data into memory at the same time

4.4.6 Availability

6-A Catch and handle exceptions where applicable
6-B Intercept the “quitting” event/key so the kid won’t accidentally quit the software.
4.5 Architectural patterns

We chose to use the Task Control Architecture Pattern for this system. The reason is that there will be a number of events that may happen independently. The software will have to listen to all of the incoming channels, and act when an event arrives.

4.5.1 Task Control Architecture Pattern

Example tasks:

The central process is the main thread that waits for events like scanning of a tag, incoming SMS and incoming phone call. The central process will then find the correct response to these events and trigger the right functionality. An example would be that a incoming SMS is received and the central process has caught the event and trigger the show SMS functionality.

4.6 Views

4.6.1 Logical view

This view shows the classes and their main external/public functions. The constructor is described unless it doesn’t have any parameters. If nothing else is mentioned, the functions are “void” (don’t return anything).
4.6.1.1 Class diagram

The most important classes are shown in this diagram. Some minor classes are left out (mainly classes that are GUI components), also not all methods and attributes are shown here, so this is not a complete class diagram. All classes and their most important functions and properties are listed below.

4.6.1.2 Classes

**Action** – This class is used to transfer information about a data tag’s actions from the Database class to other classes. This is to simplify data transfer between Database and other classes.

Constants:

<p>| int TYPE_CALL, TYPE_SENDSMS | Index constants for the different types of actions |</p>
<table>
<thead>
<tr>
<th>TYPE_SOUNDON, TYPE_SOUNDOFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>string[] ACTIONTYPES</td>
</tr>
</tbody>
</table>

Properties:

<table>
<thead>
<tr>
<th>string TagID</th>
<th>The ID of the tag, must be unique.</th>
</tr>
</thead>
<tbody>
<tr>
<td>string Name</td>
<td>The name for the action, this can be anything and won’t affect the functionality</td>
</tr>
<tr>
<td>int ActionType</td>
<td>Must be one of the action types defined by the constants in this class.</td>
</tr>
<tr>
<td>int ContactID</td>
<td>ContactID is the</td>
</tr>
<tr>
<td>string ActionString</td>
<td>Used for extra parameters for the action, for example the SMS message for TYPE SENDSMS.</td>
</tr>
</tbody>
</table>

Constructor:

| Action( string name, string tagID, int actionType, int contactID, string actionString ) | Creates a new Action class with the given properties. |

**AnimateCtl** – This is a custom image control to show animated images. It uses a timer to animate the image.

Functions:

<table>
<thead>
<tr>
<th>SetAnimation(string bmpfile, int frWidth, int DelayInterval)</th>
<th>sets the animation of the AnimateCtl, with the given frame width and delay. The image needs to be a horizontal strip of animation cells, and will be “cut up” automatically when the frame width of each cell is given. The AnimateCtl will only show the first frame until animation is started.</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartAnimation()</td>
<td>Starts the animation.</td>
</tr>
<tr>
<td>StopAnimation()</td>
<td>Stops the animation. Sets the current frame to the first frame.</td>
</tr>
</tbody>
</table>

**CallHandler** – This class manages receiving and making phone calls, and makes sure phone calls are intercepted by this application, so that the normal phone application doesn’t activate when a phone call arrives.

Properties:

<table>
<thead>
<tr>
<th>string CallerID</th>
<th>the phone number of the caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool Calling</td>
<td>A check for whether a phone call is currently active.</td>
</tr>
</tbody>
</table>

Constructor:

| CallHandler(MainGUI pClass) | “pClass” should be the parent MainGUI that uses this class for handling calls. |

Functions:

<table>
<thead>
<tr>
<th>makeCall( string number )</th>
<th>Makes a phone call to the given phone number.</th>
</tr>
</thead>
<tbody>
<tr>
<td>hangupCall()</td>
<td>Hangs up any active calls.</td>
</tr>
<tr>
<td>answerCall()</td>
<td>Answers an incoming call.</td>
</tr>
</tbody>
</table>
**Camera** – This class handles the camera functionality. It is called when a photo needs to be taken, and it calls the parent class when a photo is saved.

**Constructor:**
```
Camera( ConfigGUI parentClass )  "parentClass" should be the parent ConfigGUI object that uses this class for taking pictures.
```

**Functions:**
```
takePicture()  Starts the photo program for taking a photo. This function returns before a photo is taken, but the Camera class will call a function in the GUI when a photo is taken.
```

**ConfigGUI** – The configuration GUI for the software. This GUI is used by the user to edit contacts, data tags and set other options.

**Constructor:**
```
ConfigGUI( MainGUI maingui )  "maingui" should be the MainGUI which is the parent of this class.
```

**Functions:**
```
pictureTaken( string fullpath )  This function is called by the camera class when it has finished taking a picture, and full path is the full path to the picture taken (or null if no picture was taken).

tagRead( string tagid )  This function is called when a tag has been read while the application is in configuration mode.
```

**Contact** – This class stores info about a contact, for easier transfer of info between the Database class and other classes.

**Properties:**
```
int ID  The ID of the contact, given by the DataSet table. Must be unique.

string Name  The full name of the contact.

string Phonenumber  The phone number for the contact. The number must be checked for validity beforehand, it should only have numbers and eventually a + in front

string Picturefilename  The full path to the picture of the contact.
```

**Constructor:**
```
Contact( int ID, string name, string phonenumber, string picturefilename )  Creates a new Contact object with the given properties.
```

**Database** – This class manages all the information used by the application, and saves and loads this to a XML file.

**Constructor:**
```
Database( string filename )  "filename" should be the file name for the database file to be used for storing all
data used by this application. Should not be full path! Path to the program is added before the file name is used.

Important private functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>saveXMLFile()</td>
<td>Method to save the dataset object to the XML file given by the constructor. This function is called in each function which changes the database. If new functions are added to change the database, they should also include this function!</td>
</tr>
<tr>
<td>loadXMLFile()</td>
<td>Method for loading data from the XML file into the class. This is used only in the constructor.</td>
</tr>
<tr>
<td>init()</td>
<td>Set up the tables and columns. This is only used in the constructor. This function goes through the tables loaded from the XML file, and adds eventual missing columns to it. If no tables are loaded, it creates all needed tables.</td>
</tr>
</tbody>
</table>

Public functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact[] getContacts()</td>
<td>Returns a list of all contacts sorted by their full name.</td>
</tr>
<tr>
<td>int addContact(string name, string phonenumber, string picturefilename)</td>
<td>Adds a new contact and returns the ID of the new contact.</td>
</tr>
<tr>
<td>Contact getContact(string phonenumber)</td>
<td>This method returns a contact object when given the phonenumber.</td>
</tr>
<tr>
<td>Contact getContact( int contactid )</td>
<td>Returns the contact with the given contact ID.</td>
</tr>
<tr>
<td>editContact(int ID, string name, string phonenumber, string picturefilename)</td>
<td>Changes the information on an existing contact.</td>
</tr>
<tr>
<td>deleteContact(int ID)</td>
<td>Deletes the contact with the given ID. Also deletes any actions connected to the given contact.</td>
</tr>
<tr>
<td>Action[] getActions()</td>
<td>Returns a list of all actions sorted by their action name.</td>
</tr>
<tr>
<td>Action getAction( string ID )</td>
<td>Returns the action with the given tag ID.</td>
</tr>
<tr>
<td>bool addAction(string tagid)</td>
<td>Adds a new tag ID to the database. Returns false if the tag ID already exists.</td>
</tr>
<tr>
<td>editAction(string name, string rfid, int actiontype, int contactid, string actionstring)</td>
<td>Changes the information on an existing data tag.</td>
</tr>
<tr>
<td>deleteAction(string rfid)</td>
<td>Deletes the data tag with the given ID from the database.</td>
</tr>
<tr>
<td>int getActionCount( int contactid )</td>
<td>Returns the number of actions which is associated with the given contact ID.</td>
</tr>
<tr>
<td>setConfiguration(string key, string value)</td>
<td>Sets a key-value pair in the configuration table.</td>
</tr>
<tr>
<td>string getConfiguration(string key)</td>
<td>Gets the value with the specific key in the configuration table.</td>
</tr>
<tr>
<td>getFullPath( string filename )</td>
<td>Returns the given file name with the application path added in front of it.</td>
</tr>
</tbody>
</table>
**Fullscreen** – This is a version of the Windows Form which shows the application in full screen, without title bars or input bars. The MainGUI inherits from this class.

Most important public function:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximized()</td>
<td>This function maximizes the window, but only works once. It was separated from the loading of the class, because we needed to run this after the “evaluation” box for the TextToSpeech engine, otherwise the window wouldn’t show properly.</td>
</tr>
</tbody>
</table>

**ImageButton** – This is an image button control. (Images that are clickable.) It works exactly like a button and an image: You set the Image property to change the image, and you add a click event handler to capture button presses.

**KeyOverrider** – This class intercepts keypresses so our application won’t be exited if a hardware button is pressed.

Constants:

<table>
<thead>
<tr>
<th>Constant values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int VK_RECORD, VK_CALENDER, VK_CONTACTS, VK_ANSWER, VK_HANGUP, …</td>
<td>These are some of the most useful constants for the hardware buttons. The constants are used in the KeyPress event handler for checking which button is actually being pressed.</td>
</tr>
</tbody>
</table>

Functions:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>overrideKeys()</td>
<td>This function overrides all the hardware buttons except for the Record button, which is used by the barcode scanner. This function needs to be called again after some external programs are used, particularly the camera in ConfigGUI.</td>
</tr>
</tbody>
</table>

Important events:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPress</td>
<td>This event occurs when a hardware button is pressed. An event handler is added to this event in order to implement hardware buttons in our application. (This is used in the MainGUI to support the red and green button for answering and hanging up calls, amongst others.)</td>
</tr>
</tbody>
</table>

**MainGUI** – This is the main class of the application. This class initializes all needed classes and handles all messages, forwarding them to ConfigGUI when necessary.

Properties:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>The Database object used for storing all data used by our application.</td>
</tr>
<tr>
<td>IDBlueDriver</td>
<td>The driver for the IDBlue pen</td>
</tr>
<tr>
<td>IDBlueDevice</td>
<td>This stores the IDBlue pen device data, if it has been set.</td>
</tr>
</tbody>
</table>

The IDBlue properties and variables should be removed when IDBlue isn’t used anymore.

Functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addNewMessageData( string)</td>
<td>This function is called by the message receiver when a SMS message has been sent.</td>
</tr>
<tr>
<td>method</td>
<td>description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>messagetext, string number</td>
<td>received. If the number is in the address book, the message data is added to the queue.</td>
</tr>
<tr>
<td>messageReceived(object sender, EventArgs e)</td>
<td>This is the event listener function used for the MessageReceiver when a message is received. It shows the SMS message if the application is in standby state.</td>
</tr>
<tr>
<td>showNextMessage()</td>
<td>This shows the next SMS message in the queue. Note that the SMS isn’t removed from the queue until the SMS is closed. This allows the SMS to be shown again if there is an incoming phone call while showing the SMS.</td>
</tr>
<tr>
<td>callConnected()</td>
<td>This function is called by the call handler when the call receiver answers the call, when making a phone call.</td>
</tr>
<tr>
<td>callDisconnected()</td>
<td>This method is called by the call handler when the caller ends the call.</td>
</tr>
<tr>
<td>buttonPressed(object sender, System.EventArgs e)</td>
<td>This is the event listener function used for the KeyOverrider to capture hardware button presses.</td>
</tr>
<tr>
<td>setShutdown()</td>
<td>This function is called by the ConfigGUI when it wants the application to shut down. Then when ConfigGUI is closed, MainGUI will stop all threads and exit.</td>
</tr>
<tr>
<td>setDeviceActivated(bool state)</td>
<td>This function is called by the IDBlue class when the IDBlue pen is activated (state=true) or deactivated (state=false). This is forwarded to the ConfigGUI, so the ConfigGUI can show the correct status in the connection status bar.</td>
</tr>
<tr>
<td>finishedSearching(Device[] devices)</td>
<td>This function is called by the IDBlue class when it’s done searching for Bluetooth devices. “devices” is the list of devices found, and this is forwarded to the ConfigGUI if it is currently running (which it should be).</td>
</tr>
</tbody>
</table>

The last two functions are for the IDBlue pen, and should be removed when IDBlue is no longer used.

**MessageReceiver** – This class handles receiving SMS messages. OBS: The class changes the registry so that SMS messages won’t be received by the default application, but by our application instead.

Constructor:

```
MessageReceiver(MainGUI form) | “form” is the MainGUI parent for this object.
```

**MessageSender** – This class handles sending of SMS messages.

```
sendMessage(string phonenumber, string message) | Sends the given message to the given phone number. OBS: The function adds +47 to the phone number if there is no plus sign in front of the phone number! The sending didn’t seem to work when there were no land code in front of the phone number.
```

**PasswordBox** - This shows a small windowpanel with an inputbox for writing a password. It is possible to set the box to hide after a given amount of seconds.

Properties:
string Password | The password entered into the password text field. Returns “” if Cancel was pressed.
---|---

### Functions:
- **ShowDialog( string message )** | Shows the password box with “message” over the password field.
- **ShowDialog( string message, int seconds )** | Same as above, but the password box disappears after the given amount of seconds.

### Sound – Class to play sound clips

#### Constructors:
- **Sound(string fileName)** | Constructs a Sound object with the given sound file (must be full path!)
- **Sound(Stream stream)** | Constructs a Sound object with the given stream as sound data.

#### Functions:
- **Play()** | Plays the sound data in the sound object.

### TextToSpeech – The C# wrapper for the TextToSpeech library.

#### Functions:
- **TTS_Startup()** | Starts the TextToSpeech engine. Necessary for the engine to work.
- **TTS_Speak( string data )** | Reads the given string data.
- **TTS_StopSpeaking()** | Stops reading, if it has started.
- **TTS_Shutdown()** | Shuts down the engine.

Extra classes for using the IDBlue RFID pen. These classes may be excluded if another reader is used. Note that for the MainGUI class, we built the IDBlue functionality directly in the class, so it’ll have to be removed from the class manually.

### IDBlue – This class is for the IDBlue pen. It has a set of event handlers that calls functions in the parent class, the most important one being the event handler for when a tag is read.

#### Constructor:
- **IDBlue( MainGUI maingui )** | “maingui” should be the MainGUI parent that uses this class.

#### Functions:
- **startInit()** | Initializes drivers.
- **Dispose()** | Turns off autoconnect before disposing itself.
- **startSearchDevices()** | Starts searching for devices. When it’s done searching, the results are sent to the parent class’ finishedSearching() function.
- **setDevice()** | Sets the Bluetooth device which should be the IDBlue RFID pen, and starts autoconnecting to it.
**ConfigGUI_IDBlue** – This is an extension of the configuration GUI which works specifically with the IDBlue RFID pen.

Constructor:

<table>
<thead>
<tr>
<th>ConfigGUI_IDBlue( MainGUI maingui )</th>
<th>“maingui” is the parent MainGUI which uses this class.</th>
</tr>
</thead>
</table>

**Functions:**

<table>
<thead>
<tr>
<th>finishedSearching( Device[] devices )</th>
<th>This function is called when a list of devices has been given by the idblue driver.</th>
</tr>
</thead>
<tbody>
<tr>
<td>setDeviceActivated( bool state )</td>
<td>This function is called from the MainGUI when the IDBlue pen activated state changes.</td>
</tr>
</tbody>
</table>

### 4.6.2 Process view

This part of the architecture describes how the application runs, basically which order things are done in different situations.
### Activity diagrams

<table>
<thead>
<tr>
<th>Send SMS</th>
<th>Receive SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Send SMS Diagram" /></td>
<td><img src="image" alt="Receive SMS Diagram" /></td>
</tr>
</tbody>
</table>

The application starts in standby state, waiting for events. When a tag is scanned, the tag, contact and text message information is queried from the database. When everything is gathered from the database, and the kid has pressed the green button, the text message is sent. The application will return to standby mode.

When a text message is received, the application stores the message in a queue. Then it checks if the application is currently in standby state. This is to see if it can show the message at the receiving time or if it has to wait until the state is back to standby.

The application fetches the phone number from the SMS and checks it with the database. If the contact exists the message will be shown to the user. If not, the message will be remove from the queue and the application returns to standby state.
<table>
<thead>
<tr>
<th>Make phone call</th>
<th>Receive phone call</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="chart1.png" alt="Flowchart" /></td>
<td><img src="chart2.png" alt="Flowchart" /></td>
</tr>
</tbody>
</table>

The application is first in standby state waiting for an event. When a tag is scanned, the tag ID is checked with the database, the contact information is retrieved and the information is shown on the screen. The phone call is then made when the child presses the green button. The application returns to standby state when the conversation ends.

The phone call will be received in every state. The application checks the caller's phone number with the database and fetches contact information. If the contact information exists in the database the call will be shown to the user. If the contact does not exist, the application will take no action.
4.6.3 Development view

This view describes roughly how the development will be structured, and how the system is layered.

<table>
<thead>
<tr>
<th>Main GUI</th>
<th>Config GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messagesender</td>
<td>API</td>
</tr>
<tr>
<td>Messagereceiver</td>
<td>Database</td>
</tr>
<tr>
<td>Callhandler</td>
<td>Camera</td>
</tr>
<tr>
<td></td>
<td>Phone OS</td>
</tr>
<tr>
<td>Phone hardware</td>
<td>Scanner hardware</td>
</tr>
<tr>
<td></td>
<td>Phone memory</td>
</tr>
<tr>
<td></td>
<td>Built-in camera</td>
</tr>
</tbody>
</table>

**Yellow**: Classes from our application

**Blue**: Written and imported application programming interfaces

**Red**: Phone operative system

**White**: Hardware
5 Development

In this part of the document we will give a description of the application we have developed, together with pictures from the application. Differences from some design tactics and the reasons for those are described in the chapter 6.

5.1 User mode

When starting the application, this mode will be activated and the application will go into a standby state. When actions are activated by scanning a tag, the application will change to one of the other screens showed in the subchapters of this chapter. As shown in Figure 8: Standby state, we chose to have a blank page indicating standby state. Not having any graphical components in the user mode will prevent the child to access functionalities unintentionally, taken in account that we have a touch screen. The plain screen will also make the transition to any other functionality much more apparent. The need of using the stylus to interact with the components in this mode is not necessary, and we have chosen to make the components fit a finger press.

5.1.1 Sending SMS

When a tag is scanned, the application will check to see if it recognizes the ID in the database. If the tag ID is found and correctly configured to send a text message, the application will present a picture of an envelope of the screen, with an animation of an arrow pointing to the photo of the one who will receive the SMS. This is to demonstrate that the message in the envelope and going to that person. The message text is available for the children to read, and for those who are not able to read, they may use the text-to-speech button that will read the text out loud. When the child decides to send the text message, the green hardware button has to be pressed and if the message is sent without any errors, a green thumbs-up picture is presented on the bottom of the screen. If errors occur a red thumbs-down picture will be presented. The child may press the green hardware button once more to try and resend the
message. Finally, the red hardware button has to be pressed to close the send text message screen and return to the standby state.

5.1.2 Receiving SMS

When a text message is received by the application, it will first check if the sender’s phone number is stored in the database. One of the requirements of the application is that it should not allow communication with others than the ones stored in the database. If the number is in the database, the application adds the message to the queue and checks if it is in standby state. If it is, the message will be shown to the child immediately. If not, the application will show it to the child once the state changes back to standby.

When the message is shown, a picture of an envelope will appear, with an animation of an arrow pointing from a photo of the sender towards the envelope. This is to show that the message is from that person and is now ready to be read. The application will notice the child with a sound to attract attention. The text message is available to the child in the textbox area and with the possibility to press the ear picture button to access the text to speech option. The child has to press the red hardware button to close this window and to back to standby state. If the child forgets to close the window the alarm keep sounding to remind the child that the window is not closed.

5.1.3 Making phone calls

When a tag is scanned, and that tag is setup to make a phone call, the application will check the database and retrieve the needed information and show it to the child. A red picture of a phone and a photo of the person that will receive the phone call will be shown on the screen. Once the child decides to make the call and the green hardware button is pressed, the red phone picture will change to an animation.
When the call has been received and initialized, the phone animation will then change to a
green phone to indicate that call is connected. The red hardware button can be pressed
whenever the child wishes to disconnect the call.

5.1.4 Receiving phone calls

When a phone call is received, the application will check to see if it
recognizes the caller’s phone number in the database. If the number
is recognized it will show an animation of a calling red phone and a
picture of the caller. The application will notice the child with a
sound to attract attention. To accept the call, the child has to press
the green hardware button. When the connection is established the
red picture phone will change colour to green. To disconnect the
child can press the red hardware button.

5.2 Configuration mode

To get a functioning phone, the guardians of the child have to make some configuration.
Without any data stored into the phones database, like contact and actions, nothing will
happen on the phone when the child uses it.

5.2.1 Entering configuration mode

To enter configuration mode, the parents need to enter a button
combination using the red and green phone buttons. After
entering the code, a password box will show up, and they will
have to enter the correct password to enter configuration mode.
There is a 30 second timer on the password box, and the box will
disappear after the set amount of time, to avoid it cluttering the
screen forever if the kid accidentally enters the correct button
combination to show the password box.

The password can be set once they enter the configuration mode.
If there is no password set when they enter configuration the first
time, they will be asked to enter a new password.
Note: The password entered is shown in plain text. The reason is that the parents may not be used to entering text on such a tiny device, and obscuring the password would make it more difficult for them to see if they’ve typed the password correctly.

The password is not stored with any kind of encryption either. The purpose of the password is just to make it harder for the kids who use the phone to enter configuration mode by accident. Since all configurations for the application is stored in plain text in a XML file, the password wouldn’t really help much against outsiders even if it was encrypted.

5.2.2 Contacts

The first screen in the configuration mode allows the parents to add and edit contacts. To add a new contact, “New contact” is clicked, and the fields will be blanked out. When they click “Save” after filling in the details for the new contact, a new contact will be added to the contact list. The phone number will be checked before saving, to make sure it’s valid. If it’s not, an error message will be shown.

To edit a contact, they select the contact from the list. The contact’s details will be shown, and the parents can then change the details and click save when they’re done. They can also add a picture by clicking on the camera button. This will open the camera application, and after taking a picture it will be saved with the contact, so that it shows up when this contact is selected.

Removing the picture is done using the X button next to the camera button, and deleting contacts is done with the “Delete” button. Both these buttons will show a confirmation box before deleting, to make sure the parents won’t delete anything by mistake.

5.2.3 Actions

In our application, each data tag is associated with a specific action, for example making a phone call or sending a text message.
Adding actions is mostly done in the same way as adding contacts. However, there is no “New” button: To add a new action, the parents first have to scan in a tag ID. The ID number will be checked, and if it doesn’t exist in the list already, a new action will be created and added to the list. The parents can then edit the action to change the tag’s action, which contact the action should apply to, and eventually writing the text message that should be sent.

If a tag ID that is scanned already exists in the list, that action will be selected in the list, and the tag’s details will be shown, so they can edit the action for that tag. The parents can also show a tag’s details manually by selecting it from the list.

Deleting an action is done by clicking the “Delete” button. A confirmation box will be shown before deleting. Also note that when deleting a contact, all the actions associated with that contact will also be deleted. This is to avoid actions pointing to contacts that don’t exist.

5.2.4 Options

We haven’t added many options in our application, since it’s not really the purpose of our project. If the application is a success, and is to be worked on after this project, other options could possibly be added here.

Currently it’s only possible to turn sound on and off, and to change the password for entering configuration mode. Setting the password doesn’t require the parents to enter the old password first. (See the note on password security in the first subchapter.)
5.2.5 ID reader

The ID reader tab lets the parent set up the IDBlue pen, if they have one. Since the IDBlue pen is a Bluetooth device, it can’t be used unless the phone has created a Bluetooth connection to the pen. This only needs to be done once.

To create the connection to the IDBlue pen, they’ll first have to search for Bluetooth devices. After searching, a list of all found Bluetooth devices will be shown. The parents should then be able to select the IDBlue device from the list, and then select “Set device”, to set it as the IDBlue device to be used with the application. Windows CE will then show the “Connect to Bluetooth device” dialogs, where the user can connect to the Bluetooth device.

After the user has connected to the IDBlue pen, the application will always automatically try and connect to it every time it starts, unless the device is removed by clicking “Remove device”.

5.3 Encountered issues

The most major issues we had while developing the application was to get the phone functions to work flawlessly. Since the code for the phone functionality isn’t quite as straightforward as other parts of the code, it was a bit difficult to understand some of the problems we encountered. The main problem was the fact that receiving phone calls didn’t seem to work properly after making a phone call. Sometimes the phone just ignored the incoming phone call, other times it seemed to hang while receiving it. In the end it seemed like we had to reinitialize parts of the phone handling class after making a phone call, in order for phone call receiving to work better. We’re still not sure if it’s completely fixed, however.

There was also the issue with the RFID scanner we had available not being practical enough. We used a Bluetooth RFID pen, which wasn’t very practical because:

1. It required the users (kids with Down syndrome) to keep tab on two different objects.
2. It had quite low battery time. (We were told the battery time was two hours.)
3. If the pen had gone into hibernation mode, it required one click to turn on the pen, and then another click for reading the tags. Both clicks give the same beep feedback, so for kids it may be difficult to understand the difference, and they wouldn’t understand why the pen wouldn’t read anything.

If we had a RFID scanner similar to the barcode scanner we had (in other words, a scanner which can be attached to the phone), it may have been more practical. However, there would still be the issue with the RFID scanner not being able to scan tags when they are put on top of each other. Since the thought idea was to have the tags inside a book/booklet, the tags would most likely be on top of each other, making the RFID scanner useless.

Another issue was the Text to Speech engine we used, which is an evaluation copy. The engine shows a popup window with info about the evaluation of the engine. In itself the popup window was not that much of a problem. However, when it showed up while loading our application, it seemed to cause the window for our application not to show up properly. It seemed to work when we initialized the engine when it was run the first time, instead of initializing it while our application loaded. We’d however rather have it initialize during/right after loading, so that the parents can click on it instead of the child. We ended up setting a timer for loading the engine, so that it didn’t show the evaluation box before our application window had fully loaded.

A minor issue was to get the barcode scanner to work with our application. Since text/IDs read with the barcode scanner are fed to the Pocket PC as text input, we couldn’t just fetch the scanned ID as a string. In user mode it wasn’t much of a problem, since there wasn’t any other text input, so any text input could be assumed to be a barcode. In configuration mode, however, we’d have to try and differentiate between text input from the barcode scanner, and text entered manually from the user. We managed to get somewhat of a workaround for this, but there may still be some possibilities that users can enter text into the ID field, which is not really wanted.

6 Evaluation

This chapter will introduce the results and evaluation of the project. The application and its architecture in chapter 4 will be compared with the requirements and to what degree our
architecture supports the requirements stated in chapter 3. The research goals and methodology are also evaluated.

### 6.1 Evaluation of research methodology

The engineering method that we described in chapter 1.5 seemed to work quite well for our project, since the steps fitted well with our goals. The problem that is covered by our project can be characterized as a typical engineering type of problem:

1. There is a need
2. The need is researched
3. Something is developed to meet the need
4. The developed product is tested to see if is successful in meeting the need

In our situation the steps are:

1. Kids with Down syndrome can’t use normal cellphones
2. Which technologies can be used to help kids use a cellphone?
3. Develop a cellphone using the chosen technologies
4. Test the developed product

We therefore feel that this research method was good choice for our project.

### 6.2 Evaluation of the architecture

This part describes which of the functional and quality requirements were covered in our architecture, and how they were fulfilled in the development of the application.

#### 6.2.1 Functional requirements for the users (children)

Requirement 1, the application should be able to store tag data and connect those to specific actions, is covered. In requirement 1.1 it is stated that the application should send a predefined SMS, and it had three alternatives for choosing who to send to. With our supervisor Steinar Brede we concluded that requirement 1.1.1 “(to specific person(s))” was the best solution out of the three, so we chose it and excluded the other alternatives (1.1.2 and 1.1.3) in our application.
In requirement 1.2 it is stated that the application should make a phone call by choosing from three methods. We decided here to choose the requirement 1.2.1 (to specific person(s)), for the same reason as stated in the previous requirement.

Requirement 2, “the application should also be able to receive communication”, and requirement 2.1, “the application should receive a phone call”, are covered.

Requirements 2.2, “receiving a SMS”, and 2.2.1, “the SMS should be deleted after it has been read”, are both covered. Requirement 2.3, “the application should receive a MMS”, is not covered.

Requirement 3, “the application should have an address book to store its contact”, is covered. For requirement 3.1, “the application should identify the person who is calling by choosing between two methods”, we decided to cover the requirement 3.1.1, “by showing a picture”, and not cover 3.2.1, “reading the name with text-to-speech”. Requirement 3.2.1 can however quite easily be implemented in a future version of the application.

In this section almost all requirements were achieved, except for 2.3, “Receiving MMS”. This requirement is however not critical for the functionality of our application, and this functionality was already excluded in the prestudy. We’ve still left it in the requirements because it is still a very useful function, and in a future project it’d be good to see if it is possible to implement it at that time. Basically, all the most necessary functionality for the application has been added.

### 6.2.2 Functional requirements for the parents

Requirements 4.1 to 4.7 are all covered besides requirement 4.4 which is not completely covered, since the requirement 5.4.3, adding voice clip for the contact name, has not been covered.

Requirement 4.4.3 is not really needed, because Text-to-Speech has been added and can be used for reading the name, which would be a better solution because the name would then be more easily used in a sentence.
6.2.3 Quality requirements

The quality requirements results are based on our own testing when finishing the application.

Usability

Scenario A is deciding between options. The kid should be able to find the correct option within ten seconds, and a maximum of two menu screens to get to it (optimally only one). Using quality tactic 1-B we ended up implementing only one screen of information for each action, and there are in each screen a maximum of three options: confirm, cancel, and sometimes an option for listening to a text message. Since all the options are set up the same for each action, it should be easy for the kid to learn which button does what.

Scenario B deals with activating correct menu option. The application should active the correct menu option and an error margin should be 0%. There are only two situations which this scenario applies to:

1. When the kids scans a tag for doing an action
2. When the application is currently showing an action, and the kid needs to choose one of the options mentioned in the above scenario.

For 1, scanning a tag will always show the correct action, when the application is in the standby state. For 2, we have used the following quality tactics:

1-B a (Large graphical buttons): For the button that is used to listen to the text message, we have used a large “Ear” image for the button, which should be easily pressed with the finger. 1-B c (Only two menu choices): We always use the green and red hardware buttons for confirming and cancelling actions, so there are usually only these two buttons the kids need to remember how to use.

The buttons that are in use are not so close that it should be possible for the kids to accidentally press a button next to the one that they actually want to press.

Scenario C describes adding new and editing existing tags and actions. It should not take more than one minute to add a new tag or edit existing one. This excludes the time it takes to write eventual text message. While we didn't have any quality tactics that covered this scenario, we have tried our best, in cooperation with Steinar Brede, to make the interface user friendly. Our testing revealed that it take more time than 1 minute the first time when one is
not used to the graphical user interface, but after that it should be easy enough for the users to do it within one minute, so the requirement is covered.

**Modifiability**
Scenario D is about using different phone models. No parts of the graphical user interface should be cut out, and there should not be large blank unused spaces. We chose not to follow this scenario because most Pocket PC phones use the same screen resolution, so the application should work on them. The ones we’ve tested are Qtek s100 and Qtek 2020.

Scenario E describes the use of different scanning readers. No other parts of the code should need to be changed besides the reader class. We implemented the RFID reader first and then added the bar code reader. When implementing the barcode reader, we realised that this scenario could only be achieved when the new scanner has an API which can return the scanned value. But because the barcode reader worked in a completely different way, we had to make changes in the GUI for the barcode reader to work. This scenario is therefore not achieved for “all” scanners, but it should be achieved for the scanners that work roughly the same way as the ones we’ve implemented.

Scenario F is about independence between parts of the code. Less than 10% of other code should be affected. For this scenario we’ve followed quality tactic 2-A. We have divided the code into classes and done our best to avoid dependability. This quality requirement is difficult to measure, so we will leave the conclusion for this requirement undetermined.

**Availability**
Scenario G is about uptime. The uptime should be 100%. For this scenario we have the quality tactics 6-A and 6-B. While we have not been able to test the uptime fully, the application has not been stable enough to reach 100% uptime. We’re uncertain of the actual uptime, but it is certainly not 100%, so this scenario was not fulfilled. We suspect this is because the code for phone functions is not stable enough.

Scenario H says that the children should not be able to exit the application. There should be 0% chance of them exiting by accident. Quality tactics 6-B was followed for this scenario. More specifically, we disabled all hardware buttons from starting other programs, and we’ve
put the application in full screen, so that there aren’t any quit buttons that the children can click on. From our testing this quality requirement is absolutely covered.

**Testability**
Scenario I is about determining error location. Locations of error should be determined within 5 minutes. Because we have mostly fulfilled scenario F, finding error locations was usually not a problem. Whenever an error occurred while we were testing the application, we kept the time limit for locating the error, and therefore this quality requirement is covered.

**Performance**
Scenario J is about showing results of menu option immediately. There should be results shown within one second and if it takes more time a “loading” message should be displayed. Loading message is not implemented in the application since no function takes more than one second to execute. Following quality tactic 5-A, the application only loads images when they are needed. We’ve also kept file sizes low for the graphics and other resources used by the application, so they don’t take long time to load. This quality requirement is therefore covered.

**Security**
Scenario K describes that the children should not be able to enter the configuration mode. There should be less than 0,1% chance to do so. The configuration mode is only accessible to does who first know the key combination and then a password to enter the mode, as described in quality tactic 4-A. Therefore there are two security measures taken and it should be nearly impossible for the children to access it. This quality requirement is therefore covered.

**6.3 Evaluation of the goals**
We started this project with one main goal and three secondary goals. In this chapter we’ll evaluate and discuss which of these goals have been fulfilled and in what degree they have been fulfilled.

**6.3.1 Secondary goal G1**

“Do research on technologies which can be used for our purpose, and select one (or more) to use in our application.”
When we started working on this project, we were mostly focusing on using RFID as a scanning technology. After further research we found two other technologies which may be possible to use. We tested all three technologies in order to find out which of them was most appropriate for our application. We also thought of a lot of technologies that we could use in the phone, even though we didn’t use all of them.

All in all, we’re rather satisfied with the information we’ve been able to gather from our studies of the technology. We’ve been able to pick out the technologies which were useful, and leave out the ones which would be more work than they’re worth to implement. We feel that this goal was achieved quite successfully.

6.3.2 Secondary goal G2

“Plan and develop the application.”

Planning the application went without problems. While implementing, however, there were some issues as mentioned in chapter 5.3, but there was nothing that was a major setback for the project. The biggest problem with the application is that it is still not 100% stable. As mentioned in that chapter, we suspect this may be because the phone functions aren’t quite stable. Unfortunately, we don’t have enough time in this project to fully test for the reasons for the instability.

Still, we have been able to plan and develop a working version of the application, and we therefore count this goal as achieved.

6.3.3 Secondary goal G3

“Have users from our target group test the application, to see if it does help the kids.”

Because of time limitations, we’ve not been able to have a user test of the application. There is therefore no way for us to determine if this goal was reached. Regrettably, this also means that we won’t know if the main goal of the project was achieved.

A user test for our application would probably have taken several weeks to carry out, because the children would need time to get used to the application. Since implementation of the application was slightly delayed, we didn’t have enough time left to do this testing. A hurried
user test would probably affect the test results negatively, because the feedback would have to be gathered before the users got used to the application. Additionally, we only have two cellphone devices for testing, which would cause a hurried user test to give even less accurate information.

Hopefully, a user test can still be done in the near future, although the results won’t be part of this project, but of a future project to further work on the application we’ve developed.

7 Conclusion

The project has been a good experience. The prestudy, planning and development stages all went quite satisfyingly, and we feel that we’ve been able to develop a good product. While this wasn’t written as a goal, we’ve been able to use the scanning technology in a personal product. This is an area where the technology hasn’t been commonly used, and as mentioned in chapter 1.2, this was one of our motivations for this project.

Another point that we discovered in our project, was that RFID wasn’t quite as appropriate for our project as we’d expected. Because of the problem with RFID tags not being readable when they were put on top of each other, even with a layer of paper between. Therefore the tags would not be possible to use in a notebook, as described in the scenarios in chapter 1.3. Still, RFID was possible to implement, and for other types of usage could possibly be a better choice.

Even though we implemented both the RFID scanner and the barcode scanner, the scanning technology that we ended up preferring was the barcode reader, which is an older technology than RFID. The barcode reader worked sufficiently easy enough, and didn’t have the same problem as the RFID reader, where the tags couldn’t be stacked. This does not necessarily mean that this technology is better, just that it is more appropriate in our case. In addition to this, using barcodes is actually a cheaper solution, because anyone can print their own barcodes. The children also know the barcode technology from seeing it in use at supermarkets, so it may be easier for them to learn how to use it. All in all, it was rather surprising that this old technology would work so well with our application.

Our biggest regrets are that we didn’t get enough time to test the product, and the fact that the product isn’t fully stable. While we think that our lack of domain knowledge about
programming phone functionality may have caused the instability, we suspect that the phone itself may have contributed to the problem. It may have helped if our application was developed as a replacement OS.

It is regrettable that we won’t know if our main goal was reached, but we still feel that we have accomplished a lot in our project. We also feel that the application would be a help for children with Down syndrome, and maybe for other people with different disabilities as well, although we haven’t taken this into consideration in our project. We hope that a future project would be able to make a 100% stable version of our application.

8 Further work

An important aspect of any further work on our project, would be to organize a user test of the application we’ve developed, eventually make the application more stable first. The most important goal of the test would be to see if the application at all is any help for the children with Down syndrome. Another goal would be to see what the users felt were good and bad with the application, in order to further improve it.

Since the main focus of this project was to see if our proposed application was of any use to the target group, we have focused mostly the core functionality rather than implementing as much functionality as possible. There were also some functionality that we had originally thought about adding, but which we ended up leaving out because of technical difficulties of implementing it, and the limited time for our project.

The following list describes the functions that we thought of adding, but didn’t have time to add:

- add an option to set specific time periods to turn phone on and off. This would for example allow parents to set the phone to soundless mode when the kids are at school.
- make it easier to change the look of the application, so the parents can use graphics that the children are familiar with
- make it easier for the parents to change the ringing and text message sounds
- make it possible to use Text-to-Speech instead of or in addition to ringing signals when there are incoming calls/text messages. The phone could say something like “You have a phone call from (name)! Pick up the phone!” when there is a phone call, instead of just playing a ringing sound.
Also there were the functions that we didn’t add because they were too demanding for our project, but which may be relevant in a future project:

- sending and receiving MMS
- Text-to-Video (“translate” text messages into a video clip which shows the text with sign language)
- Text-to-Symbols (“translate” text messages into a set of symbols)

9 Glossary

Amblyopic

One who has amblyopia (diminished vision without structural abnormality of the visual pathway).

API – Application Programming Interface

A library or set of libraries that allow programmers to implement hardware or software solutions in their applications.

Bluetooth

A technology for wireless interaction between Bluetooth devices.

Down syndrome

A term used for a number of genetic disorders which causes a person to have learning difficulties and also physical disabilities.

ECMA – European Computer Manufacturers Association

An European association for standardising information and communication systems.

EIT – Ekspertet i Team (Interdisciplinary team work)

A project assignment for students at master level at NTNU, where the main purpose is to have students from different branches of study to work together.

GPRS – General Packet Radio Service

A transmission service commonly used by cellphones.

GUI – Graphical User Interface

The graphical aspect of an application, which presents the information and results from the application to the user, and allows user input.

MMS – Multimedia Messaging System

A system of transmitting not only text messages, but also various kinds of multimedia contents (e.g. images, audio &/or video clips) over wireless networks using the Wireless Application Protocol (WAP) protocol.
NFC – Near Field Communication
This technology is used for very short range two-way wireless connectivity, and uses a short-range radio frequency (RF) technology that allows a reader to read small amounts of data from other devices or tags when brought next to each other.

NTNU – Norges teknisk-naturvitenskaplige universitet (Norwegian University of Science and Technology)
A university located in Trondheim, Norway.

PDA – Personal Digital Assistant
A small handheld device which is basically a computer with limited resources, usually with a touch screen for interaction.

RFID – Radio Frequency Identification
An automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags.

SMS – Short Message Service
A cellphone service which lets users send short text messages to other cellphone users.

SOA – Service Oriented Architecture
SOA expresses a software architectural concept that defines the use of services to support the requirements of software users. In a SOA environment, nodes on a network make resources available to other participants in the network as independent services that the participants access in a standardized way.

SOAP
SOAP is a protocol for exchanging XML-based messages over a computer network.

Stylus
A pointing device roughly shaped as a pen used for interactions with a touch screen.

Symbian
A set of operating systems created in a cooperation between several cellphone companies, amongst others Nokia and Sony Ericsson.

WSDL – Web Services Description Language
An XML format published for describing Web services.

XML – eXtensible Markup Language
A general-purpose markup language for creating special-purpose markup languages, capable of describing many different kinds of data.
10 References


[12] Craig Neable, *The .NET Compact Framework*


[19] Qtek, [http://www.qtek.no](http://www.qtek.no)


[22] IDBlue Bluetooth RFID pen, [http://www.cathexis.com/Products/idblue.asp](http://www.cathexis.com/Products/idblue.asp)


### 11 Appendices

#### 11.1 Specifications of Qtek s100

**Platform**
- Small PDA form factor integrated GSM/GPRS, Bluetooth, and 1.3 megapixel camera
- Dimension: 108 mm(L) x 58 mm(W) x 18.1 mm(T)
- Weight 155 g (with battery)

**Memory**
- ROM: 64 MB
- RAM: 64 MB SDRAM

**LCD Module**
- 2.8” 240 x 320 dots resolution
- 64K-color TFT Transflective LCD
- LED back light
- Sensitive Touch Screen

**Processor/Chipset**
- Intel Bulverde 416 MHz

**GSM/GPRS Functional Block**
- Internal antenna
- GPRS / GSM (Tri-band) module:
- GSM900: 880-915, 925-960MHz
- DCS1800: 1710-1785, 1805-1880MHz
- PCS1900: 1850 -1910, 1930-1990MHz

**Camera**
- Colors CMOS 1.3 mega-pixel camera

**Device-to-Device Connectivity**
- Bluetooth
- Compliant with v1.2
- Class 2 transmit power
- Infrared IrDA SIR

**Platform**
- Small PDA form factor integrated GSM/GPRS, Bluetooth, and 1.3 mega-pixel camera

**Physical descriptions**
- Dimension: 108 mm(L) x 58 mm(W) x 18.1 mm(T)
- Weight 155 g (with battery)

**11.2 Nokia 5140 Technical Specifications**

**Operating System:**
- Nokia OS

**Developer Platform:**
- Series 40 Developer Platform 2.0

**Java Technology:**
- CLDC 1.1
- Wireless Messaging API (JSR-120)
- Mobile Media API (JSR-135)
- JTWI (JSR-185)
- MIDP 2.0
- Nokia UI API

**Browser:**
- WAP 2.0
- XHTML over TCP/IP

**Messaging:**
- MMS+SMIL
• SMS

UAProfile:
• Profile 1

Digital Rights Management:
• OMA DRM v1.0

Delivery Method:
• MMS
• WAP Download

Sound Formats:
• AMR (NB-AMR)
• MIDI tones (poly 16)

Band Functionality:
• GSM 1800
• GSM 1900
• GSM 900

Regional Availability:
• Africa
• Americas
• Asia-Pacific
• Europe

Screen Display:
• Color Depth: 12 bit
• Resolution: 128 x 128

Physical Descriptions:
• Dimensions: 106.5 x 47 x 24 mm
• Weight: 101 g

Memory:
• Heap size: 500 KB
• Shared Memory for Storage: 977 KB
• Max JAR Size: 125 KB

Keypad Descriptions:
• 3 labeled soft keys
• 4-way scrolling
• Grid key mat

Local Connectivity:
• Infrared
• USB

Video Support:
• 3GPP formats (H.263)

Network Data Support:
• HSCSD
• CSD
• EGPRS
• GPRS

PC Connectivity:
• Infrared
• USB

Extra Features:
• Flashlight
• FM radio
• Handsfree speaker
• Instant Messaging
• Thermometer
• VGA camera