Software Engineering, Aalborg University

Sixth Semester Project, 2011

F11s601a

DigiPECS

Students:
Mette BANK
Kenneth BRODERSEN
Rikke JENSEN
Thomas PEDERSEN

Supervisor:
Ulrik NYMAN

May 26, 2011

Cassiopeia - Department of Computer Science
Selma Lagerlöfs Vej 300, 9220 Aalborg Øst

Title:
DigiPECS

Area:
Application Development

Field of Study:
Bachelor in Software Engineering

Project Period:
1st February 2011 - 27th May 2011

Project Group:
f11s601a

Participants:
Mette Bank
Kenneth Halberg Brodersen
Rikke Hagensby Jensen
Thomas Pedersen

Supervisor:
Ulrik Nyman

Number of Printed copies:
8

Number of Pages:
139

Date of completion:
May 26, 2011

DigiPECS is a digitised version of the Picture Exchange Communication System (PECS), primarily targeted at children with autism. DigiPECS is a part of a multi project developed across four groups with each distinct responsibility, aiming at making a protected environment for children with limited mental capabilities.
We demonstrate DigiPECS using a Samsung Galaxy Tab running Android 2.2, and implement it as an application for the unified system GIRAF. We evaluate DigiPECS in more aspects to assure high usability, high quality code and reliability and robustness of the software. The results obtained through testing as well as and an evaluation of the system compared to the stated requirements is presented.

PREFACE

This report is the result of a sixth semester project made by four software engineering students, group f11s601a, at Aalborg University during the spring 2011.

This particular project is our contribution to a multi project developed jointly by four groups, and documents the analysis, design, implementation and evaluation of our part of the unified system.

The target audience of this report is fellow students who could later have an interest in making further developments to the system. Further, anyone with an interest in assistive technology mainly focused on children with autism is also a target audience.

The report is split into four distinct parts, each with different focus. First part concerns the overall system to be developed during the multi project process. Both part two and part three mainly relates to the part of the system developed within this group, though some aspects of integrating the entire system as well will be clarified. Last part is a closure, concluding on both the entire system as well as the part we are responsible for, and will also comprise proposals for future improvements.

In this report, source material references will be notated with the initial letter of the auther(s) surname followed by the year of publication. E.g. [BCK03], a book written by Len Bass, Paul Clements, and Rick Kazman, published in 2003. The source refers to an entry in the bibliography list, in which details regarding the source can be found. If a reference to a particular page in a source is needed, the source reference will be followed by a page indication, e.g. [BCK03, p.123]. A CD, containing the full source code and pdf version of this report, is enclosed.

Group f11s601a would like to thank supervisor Ulrik Nyman, Aalborg University, who throughout this project has supported with his insider knowledge, constructive feedback and assistance.

Further, we would like to thank Birken daycare centre – with a special thank to Bente Thrysøe – for contributing with knowledge, ideas and feedback during this development.

Enjoy reading!

Group f11s601a

CONTENTS

Part I

Multi Project

CHAPTER 1

EVOLUTION

Ever since the invention of the telephone it has been a target for constant evolution. From the beginning the objective with the telephone has been to communicate across distances. But with the extension of the telephone from being dependent on cables to operate wireless, from dialling to touch phones, the phone has also developed from just being a way to communicate to having many purposes. Planning, entertaining, Internet browsing, E-reader just to name a few of the functionalities added to a modern mobile phone.

Simultaneously with the development and extensions of the telephone, many initiatives are taken to aid people with some kind of disability. Many kinds of assistive technologies are available today such as hearing aid for hearing impaired, electrical wheelchairs for people who are paralysed and so forth. Also concerning mobile phones several initiatives are taken to aid disabilities. People with learning disabilities are offered e.g. spell checking and Text-To-Speech (TTS), all with the purpose of widen the market of users. However, the common denominator for these assistive technologies is that they extensions of functionality working within the same environment as for the masses.

Furthermore, today mobile phones are getting so widespread and common, that now relatively young people own their private mobile phone. The best way for parents to control the usage of the phone for their children, is by restricting the expenses on the phone.

An interesting extension of the parent control could be looking at the market of young/children who have some kind of handicap, making it essential for their parent to control usage of the unit. The control should not only encompass the expenses, but could also be the basic functionalities offered to their children. This would require a strictly protected environment for the basic use of the phone, and a protected mode where permissions for usage could be controlled – potentially in accordance with the child's capabilities. This would require a system dedicated the special target users and their exact needs. This is the base of origin and inspiration for this project.

CHAPTER 2

MULTI PROJECT DESCRIPTION

Inspired to create an entire system dedicated to children with special needs, combining an assistive technology with a mobile phone, and giving the parent of the child a control of *what* is available *when*, we need to determine not only *who* could benefit from such a system, but also *what* needs to be controlled.

This leads to more choices. Should the system aim at one kind of disability or should the system embrace users with different kinds of disabilities with united needs? Should the system be totally 'protected' for the child, giving the parent full control of what is available, or should some options be given to the child?

By choosing the platform, Android, to develop for as well as the opportunities this gives, helps deciding this. Similar to Android Market, it is possible to make a web based application store, where different applications suited for different needs can be available, making the 'supervisor' of the phone capable of installing functionality to the phone tailored the needs of a unique child. This means, that the system by default could offer a high degree of protection, meaning as little control as possible is given to the child. *But* the parent of the child can through an administration interface – well hidden from the child – install functionality to the phone, giving the child the opportunity to choose what to do, but only when allowed from the parent. This would allow for having a target group for the entire system embracing children with different disabilities, while applications to install on the system could be tailored some exact needs by a subset of the more broad target group; *children with limited mental capabilities.*

2.1 Initial Problem

Having defined an overall target group and the basic needs they have, some initial requirements for the system can be posted. We need to make a system that hides the complexity that usually is present, and instead offer the user a tool to primarily assist them in their everyday life and function as a learning tool. But as with almost all other mobile units on the market today, it should also be possible to add functionality aiming at entertainment, though with parent control. Hiding the complexity includes hiding access to the administration of the unit, hiding the use of the basic functionality of a standard phone (such as making calls, texting), and hiding all navigation that can lead the user to a place on the phone where (s)he does not know how to get back to a 'safe place'. This yields for a system that must be simple to use.

The system will be developed as a cooperation between four groups. Each group will have the responsibility of making a part of the unified system, implementing features that have emerged from common decisions.

With this in mind, some overall agreements regarding the system can be stated. These

are stated using a system definition as well as en overall requirement specification, presented in the next sections.

2.2 System Definition

Before stating the requirements of the system, some overall terminology has to be addressed. The overall system to be developed during this multi project is called *GIRAF* . GIRAF is a development name, which stands for Graphical Interface Resource for Assistive Functionality. The web-application serving as a market place is called *GirafPlace* . The parent of the child using the mobile unit, as well as other adults responsible for the child at any given time, is referred to using the collective name *guardian*.

2.2.1 FACTORS

Functionality The system should offer installation of new applications and make it possible to administrate common settings by need. The system should mask the normal functionalities of the unit to the user. Further, the system should give the opportunity to control the usage of and access to system- and user profile settings as well as applications according to the current time, and location of the unit. The system should be delivered with a number of pre-installed applications which is customizable to the user.

Application Domain Children with limited mental capabilities due to handicap or age, making it hard for them to handle the complexity of a normal smart-phone or tablet Operating System (OS). Parents and kindergarten teachers (guardians) will be in charge of administrating the system.

Conditions The project is being developed by a number of study groups as a study project, and thus has a hard deadline that cannot be exceeded. The system should be simple and intuitive to use. The system should be developed such that it is customizable to the individual child and its disabilities. Further, it should allow guardians to limit the functionality of the system. To allow other application developers to continue to develop the system and further applications after this semester, the system should be maintainable.

Technology The system must run on Android touch devices such as smart-phones and tablets. Different hardware should be supported, although it is required that the unit is running Android 2.2 or newer. The system should mainly be developed using Java and the Android Software Development Kit (SDK) version 8 for Android 2.2.

Objects A smart-phone or tablet device. The Android platform. Global system- and application specific settings. Applications.

Responsibility The system should act as an assistive tool by providing pre-installed applications developed to aid and entertain the small-aged and disabled children using the system. Further, the system should provide the opportunity to install other third party applications. Through a home menu, the system should in accordance with the location, the user profile as well as the global settings of the system control which applications the user is allowed to access.

Sub Systems An administration module should provide access to user profile properties as well as global and specific application and system settings. A home menu must provide access to applications in accordance with the location, the user profile and the global settings of the system. Pre-installed applications including a day-planning tool and a PECS application.

2.2.2 System Definition

A simple and intuitive module based single user system for Android touch devices, such as smart-phones and tablets. By masking the normal interface of an Android device, the device should offer functionality that is suitable for the intended user.

The system should be responsible for aiding and entertaining children with limited mental capabilities due to mental handicap and/or age, having a difficult time handling the complexity of a normal smart-phone or tablet OS. Guardians should be able to administrate the system by controlling selected application-, system- and user-specific settings through an administration interface on the phone.

Based on these settings, as well as the location of the unit, a home menu should be responsible for providing access to applications that conforms to the current settings and the state of the system. It should be possible for any third party to develop and provide additional applications to the system.

Beyond that, the system must be delivered with a set of pre-installed applications consisting of a visual, day-to-day, planning tool, and a PECS application. The system should be developed using Java and the recent version of the Android SDK. It is expected that the system supports Android 2.2. Further, it is expected that the system is maintainable to such a degree, that it allows other developers to keep developing the system as well as applications to the system after this semester.

2.3 Requirement Specification

The system definition has stated some overall requirements for the system to conform to. Further, some more specific agreements are elaborated. These are a mixture of non-functional attributes to design each sub-component in accordance with and some actual requirements of present features and settings in the system. In the following they are all presented as a list of requirements for the system.

Global settings Global settings should be managed through a centralised administration interface on the mobile unit. The global settings should be a blend of settings to personalise the system and filter applications according to the capabilities of the child. The settings used to filter applications should offer usage in two ways; By *developers*, to state which capabilities an intended user must possess in order to use/benefit from the application. By setting these capabilities, the application, when uploaded to GirafPlace, should only appear on the list of available applications for download synchronised with the settings on the mobile unit. Further the settings are to be used by the *guardian* of the child, to not only set a filter for applications to download, but also to set a filter locally on the phone, only showing applications to the child according to this filter. If the child looses some capability and the settings on the phone are changed accordingly, applications prior available using the lost capability will be hidden from the child.

The following global settings that should be implemented are:

- Name *(In order to personalise the system)*
- Address
- Parent's phone number
- Gender
- Birtday
- Capabilities
  - Can the child hear?
  - Can the child read?

- Can the child speak?
- Does the child know the numerals?
- Does the child know the clock? *(digital or analogue)*
- Does the child have impaired vision?
- Does the child require large buttons?
- Does the child require simple visual effects?
- Can the child use drag and drop?
- Can the child use the keyboard?

The list embraces a wide range of capabilities, that are not all going to be used by the applications developed during this project. Nevertheless, they are an attempt to capture future requirements set by application developers and guardians of children.

Installation Applications should be available for installation through GirafPlace. GirafPlace is a market place where applications developed to the system are available.

Administration The administration interface should be protected and inaccessible for the child using the mobile unit.

Maintainability The software developed should be maintainable in order for other developers to make further developments for the system. To obtain this the code should be well documented.

Graphical icon Each application developed for the system should have a unique graphical icon in the launcher making it easy for a child to recognise.

Icon guidelines All icons, both in launcer and in applications, should conform to the Icon Design Guidelines[Inc11a], provided by Google.

Easy to use Due to the specific target group, the overall system as well as each independent application should have a high degree of usability and be easy to use.

2.4 Sub-Components

Having stated the overall requirements to the complete system, each component and their responsibilities can be presented. The main system can be divided into two components, closely attached to each other, *Laucher* and *Admin*. Additionally, a *Marketplace* is developed. Further, two applications are available for the system, *DigiPECS* and *Scheduler*. A system only consisting of a launcher and an administration interface, would not be applicable for the potential users, without these applications.

A short introduction to each component will be given in the following, and in the next section the architecture and the interaction between the components will be presented.

Launcher The overall responsibility for the launcher is to make a protected environment – a sandbox – in which execution of applications can take place. This additionally comprise the functionality of adding and removing applications in this environment, as well as the responsibility to make available a way to enter global settings.

Administration The responsibility of Administration is to offer a centralised administration interface to the guardians. By entering the centralised administration through the protection offered by the launcher, the guardian will get access to the global settings of the system as well as application specific settings. When altering the global settings, the applications available locally on the phone will only be visible according to these settings. Further, when entering GirafPlace, only applications developed to children with certain

capabilities in accordance with the capabilities set on the mobile unit will be available for download.

Also, the responsibility of Administration is to offer an interface to access the application specific settings. The responsibility of handling these settings is distributed to the applications.

Marketplace The marketplace - called GirafPlace – is a web-based application store, similar to Android Market, but with modifications to assure protection for the user. Only the guardian of a child using the touch unit has access to the marketplace where available applications are shown. The responsibility of this component is to only show applications that the user could have interest in, and further to evaluate the applications available, making sure that no malicious applications could be installed on the units. The development of both the Launcher and the Marketplace lies within a single group.

Scheduler The responsibility of the scheduler is to provide an application to be available in the system, and thereby offer functionality to the users. The target group of the Scheduler is children with autism, and the aim is to aid them in planning and structuring their everyday life. The Scheduler is responsible for complying with the requirements set to the overall system and make use of the interface of the administration through the launchers protection, to access both global as well as application specific settings. The application specific settings, when accessed, is the responsibility of Scheduler.

PECS The responsibility of the PECS is to provide an application to be available in the system, and thereby offer functionality to the users. The target group of PECS is children with autism, and the aim is to assist and teach them communication skills. PECS is responsible for complying with the requirements set to the overall system and make use of the administration's interface through the launchers protection to access both global as well as application specific settings. The application specific settings, when accessed, is PECS' responsibility.

2.5 Architecture

The following will take us on a tour through the system, to discover how all the different components should interact. To keep this tour as a brief overview, some specific implementation details will be excluded. This also minimises the required knowledge about Android.

Two main types of users will be considered; a child and a guardian of the child. The guardian may of course be any other person, responsible for administrating the system. Users will refer to either the child or the guardian. We will assume that the guardian already installed the main application. This will in Android typically happen through a Market, which is an application on the unit, where applications can be installed through. The main application includes the Launcher, Admin GUI, Lib, and GirafPlace, shown in Figure 2.1.

Users will enter the system into the Launcher – this is the closed and safe environment for the child. The guardian is then capable of entering the Administration Interface (Admin GUI), this happens through a key-combination, not known by the child. The Admin GUI is the place where all administration of the system happens. First the guardian sets the general settings accordingly to the capabilities of the child, see Section 2.3. These general settings are loaded and saved through the Lib, which is an interface to the Admin database for shared/global settings. The Lib will also contain code assisting Applications in keeping the safe environment, constructed by the Launcher.

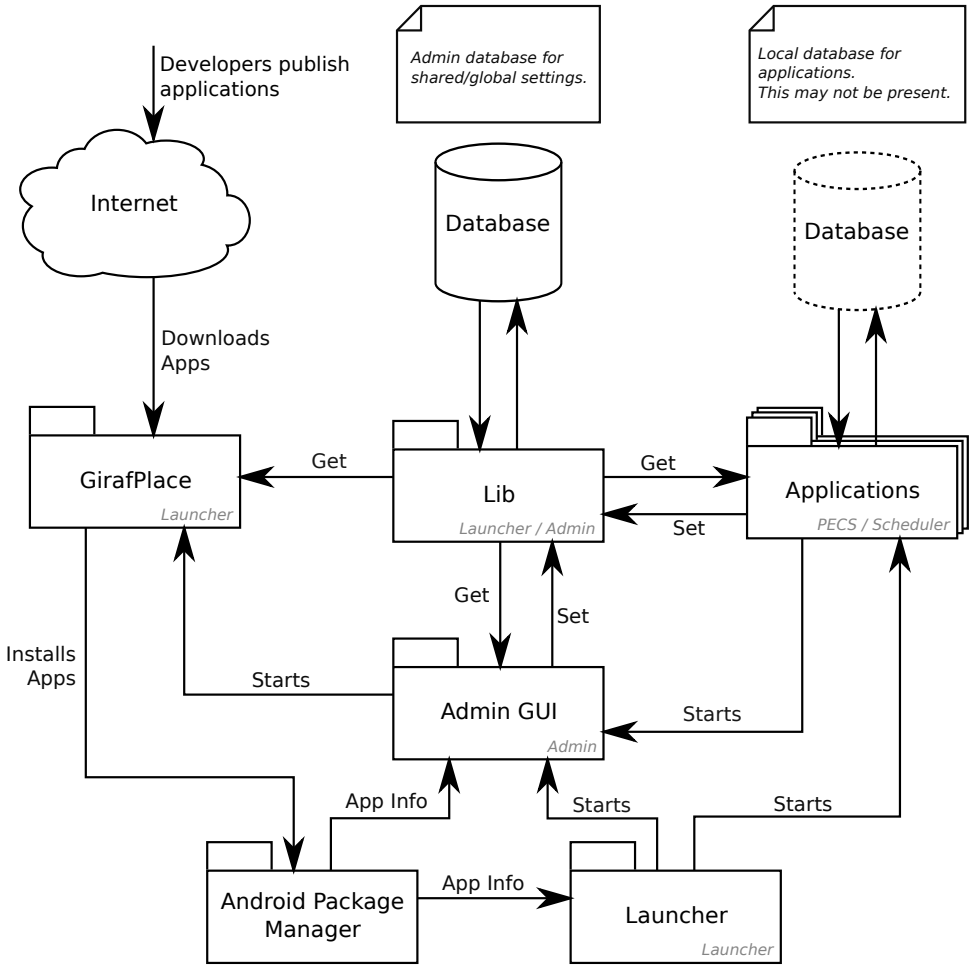


Figure 2.1: Architectural diagram for the multi project. The Android Package Manager is an included part of the Android Platform and thus not developed as part of this project. Grey text indicates the sub-component responsible for element.

From the Admin GUI the guardian may also enter GirafPlace. This is an application similar to Android Market, where the guardian can browse through a list of applications and install them. This list of applications is filtered to fit the general settings on the unit – thus GirafPlace needs to get those from Lib. When the guardian clicks to install an application, this application is downloaded from the internet, and then installed through the standard Android Package Manager – that is, the application is installed as every other Android application is.

Lets consider how an Application gets from the developer into GirafPlace, before continuing the tour. A developer publishes the Application through a website, which will upload the Application to an Internet server. In this process the Application is verified that it correspond to the specification required. If the application is accepted, it is listed on the server and can end be downloaded and installed by clients. The specification includes elements which are required for the system to correctly detect the application and allows it to be launched and administrated through this system.

Continuing the tour, the guardian then leaves GirafPlace and returns to the Admin GUI. In here, a new entry appears, representing the newly installed Application. To show applications the Admin GUI needs application information from the Android Package Manager. Through this entry the guardian may then configure the application to fit their needs. The

Admin GUI is exited and the system is now back in the Launcher. As this is the safe environment, the unit may now safely be handed to the child, which now can enter the newly installed Application. To show a list of available applications, the Launcher also requires application information from the Android Package Manager.

The Application is then launched, by the child. During the execution, it can load and save settings through the Lib. This allows the Application to act accordingly to global settings or save and restore their own. The Application may also, like any other Android application, have its own database. While the child is using the Application the guardian may desire to change an Application specific settings. E.g. the child is playing a game, and the guardian is annoyed by the sounds and wants to deactivate all sounds within the game. In this case the guardian uses the secret key combination, which starts the Admin GUI. Now the settings may be changed. The Admin GUI is left, and the unit is back into the Application.

Finally, when the Application is exited the system enters back into the Launcher.

2.5.1 Rationale

Here, we will discuss some of the underlying reasons and choices behind Figure 2.1. Constructing a safe environment is one of the key features of this system. The architecture addresses this problem by constructing a new launcher, through which applications may be launched. Constructing our own launcher allows us to filter the application shown, and thus only showing applications that we want to be accessible by the child. This means that the child cannot by default browse the Internet or make calls – this way the phone bill will be controlled. Separate applications may be constructed to safely open these features to the child, e.g. allowing them to call their parents only.

Applications need to be easy to locate and install. The typical method for achieving this on Android, is to submit the application to Android Market. However this brings up a few questions in this system; how to make a clear separation between normal applications and GIRAF application, that we intent to construct? As the children have different capabilities and needs, how can we filter applications, so that only those related to a specific child are shown?

These questions have been addressed, by constructing our own edition of Android Market – GirafPlace. This also allows us to verify applications as they are submitted by the developers, both automatic checks of file structure and also further manual checks if it is needed. Manual checks could ensure quality of the application to comply with a defined standard.

Like any other mobile unit, this system also has a need for global settings. Settings which controls how the system generally acts. These settings are administrated through the Lib. The Admin GUI provides a form to change the settings and the Lib provides facilities to access and change them in a database. The Lib is available to all applications. This means that everyone is capable of accessing these settings when needed.

2.6 Development Strategy

To coordinate the development across four different groups, some initiatives have to be taken in order to get a coherent end system. No overall development strategy is chosen, but in order to keep the system interconnected and aiming at the same objective, a representative of each group is appointed the *leader* role. With intervals between one and two weeks, the leaders have scheduled a meeting, discussing relevant matters from an agenda made by all students previously to the meeting. Minutes from the meeting are subsequently released on a shared *wiki*. Further, two *grand meetings* for all the project members have been held to make some overall decisions and to inform on status of the project.

Throughout the development the communication has been more important than documentation, which has entailed that neither a thorough list of requirements, a detailed design nor choices made throughout the development are available as documentation.

Each individual group has developed their part of the unified system using their own tailored development strategy, having the responsibility of making their component complying with the requirements made to the overall system. However, a common characteristic of all the groups is that all have been using an agile development approach.

In the latter part of the development, some tasks concerning the multi project have been distributed to sub-groups consisting of a member from each group – e.g. a sub-group was formed to test integration of the unified system.

Through discussions at leader meetings, it was decided that multi project should be developed as an open source project, allowing other developers to take part of the project and make contributions in the future. As a result the project is posted under GNU General Public License Version 3, to ensure that the software is free to use, and permitting others to distribute the code as long as it stays free to use.

To help post the multi project it was determined to utilise Google Hosting Project. Apart from providing an entrance to the project, this hosting site offers several useful features such as an SVN repository, an issue tracker and wiki page where tips, documents and development advice can be shared. The sw6 project hosting site can be found at http://code.google.com/p/sw6android/.

Part II

Pre-game

CHAPTER 3

INTRODUCTION TO DIGIPECS

The purpose with GIRAF is to aid children with a handicap in learning, performing daily tasks or being entertained by creating a system that hides the complexity of a normal mobile operating system and allows them to focus on a set of custom developed applications. The lockdown of the platform and the centralisation of the administration interface will support these goals, but the vast majority of the user's time will be spent in the individual applications. Our group has been assigned the task of developing one of these applications. It is our responsibility to create an application that integrates well with GIRAF and with a high focus on usability. To do this, knowing and understanding the intended users is important. Defining these users and their specific goals with using GIRAF is difficult. This is due to the fact that these users are children with a wide range of handicaps or perfectly healthy, but young, children. Each of these groups have their own characteristics and challenges which is reflected in their daily life and in the assistance they receive.

This difference in capabilities and challenges requires us to select a subset of the target users and develop an application with a very specific goal.

The target group for the application developed by our group is children with a form of autism. Autism is a developmental disorder that affects the brain's normal development of social and communication skills [fBIN10]. The result is often a slowly developed, or completely lacking, verbal language. Instead they often develop an increased capability to use their sight and understand the world around them using their eyes. This property has been exploited in developing a special communication system which allows an child with autism to communicate with others through the exchange of pictures. This is called the Picture Exchange Communication System or PECS for short [BF94b]. PECS is employing use of books and multiple pictures. We would like to use the ideas behind PECS and produce a digital version, called DigiPECS, which will run in GIRAF. To do this a more thorough understanding of autism and PECS is needed. This understanding is gained though research into the subject, and through the collaboration with Birken, a daycare centre for children with autism. This chapter introduces autism, PECS and gives a general introduction to the concepts of communication. The result of this chapter will be a problem definition presented in Section 3.4.

3.1 Communication

The ability to communicate is a fundamental skill to master for any human being. For most people this skill is taught from an early age, as it is a fundamental component of being sociable and feeling part of the community. But what is communication and how can it be conducted? The process of communicating requires a sender that wishes to communicate a message and a recipient for the message. In order for the process to be successful,

the recipient has to understand the message. The most common way of systematically approaching this process is to use a language. A language, will in general, define a set of rules of how to send and understand messages and a set of symbols, signs, pictures or sounds used to compose and decompose a message [WwJTO09].

Communication can be conducted in many ways such as oral, by gestures or through writing. Learning to communicate verbally through a spoken language is taught in the earliest stages of a child's development. Some individuals can have difficulties learning this ability, which may be caused by social, mental or physical impairments. Augmentative and Alternative Communication (AAC) is a set of methods and tools developed for individuals who have a verbal communication impairment [VUBK09]. Picture Exchange Communication System is a specific AAC method targeted at young children diagnosed with autism, as up to 80% of these children with the age of five display no useful verbal language [BF99].

3.2 Autism

The following section is based upon two booklets by "Videnscenter for Autisme" [JIS+06] [Sør99]. Autism Spectrum Disorders (ASD) is a term that embraces various kinds of autism. The classification of ASD in Denmark is based on The World Health Organisationt's (WHO) International Classification of Diseases (ICD-10). ASD encompasses different kinds of autism, including infantile autism, atypical autism, and Asperger syndrome. Further Other and Unspecified Pervasive developmental disorder is categorised in ASD.

ASD children primarily have difficulties in participating in emotional communication. Lack of eye contact, the ability to read body signs and for many of the autistic children the ability to verbally express their wants and needs, means that these children have difficulties learning the way "normal" children does.

The diagnose of a child with autism is primarily based on behaviour observation and can be diagnosed as early as at the age of two. The diagnose depends both on the severity of the symptoms as well as the kind of autism. The symptoms can be very evident with the child showing no interest in the surroundings at all, or they can be so vague that a clear diagnose is hard to give.

For this project a distinction between the various kinds of autism defined in ASD is not of interest, as the scope of the project is not to aid an autistic child based on a diagnose. Further, which symptoms as well as the severity of these are not primary focus. The focus is on the common characteristics of an autistic child with regards to how they communicate.

The problem of communication is alleviated by using PECS (detailed in section 3.3).

3.3 The Picture Exchange Communication System

Picture Exchange Communication System (PECS) is a functional training communication system, based on exchanging pictures. The system is designed to aid young children diagnosed with autism to communicate, by providing a tool where they can express desires, feelings and needs without having to verbalise them [BF99]. The reasoning for designing such a system is that children with autism have difficulties developing communication skills. PECS will train communication skills that are essential to have, and gives these children a way to successfully communicate with others. Studies show that although the primary goal of PECS is to provide a tool to enable functional communication skills, it is also possible that children may later develop oral abilities [BF94b].

3.3.1 Training with PECS

Training with PECS can start with children as young as two years old. It is recommended that the training involves one or more trainers, normally kindergarten teachers at the child's day care centre, that are familiar with PECS training techniques. However, anyone

that is familiar with the system can act as communication partner [BF94a]. This means that parents can comfortably adopt the system and use it at home.

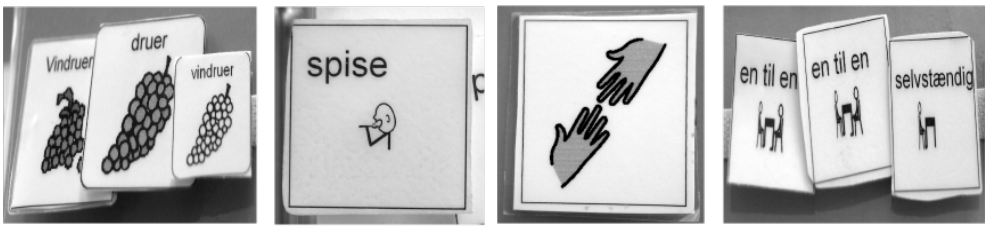
The main focus in the training with PECS is to teach autistic children to initiate communication. From the beginning the child will be taught, that it is their responsibility to approach a partner to begin communication. This is accomplished by using pictures that illustrate items that are highly reinforcing [BF99]. Such an item could be chocolate, a toy, or anything that the child desires. The child is then taught to use a picture to request this reinforcing item. So the initial task in training a child's communication skills is to figure out which items the child typically wants. The next step is to make pictures of these items and then the actual training of PECS can begin.

PECS is taught in six distinct phases, where the development level of the child is reflected in each phase [BF94b] In the initial phases the child is taught that the communication between the child and communication partner is directed through pictures. Gradually the amount and complexity of the pictures are increased and the child may even learn to use words verbally. In the middle phases the child is introduced to constructing sentences with pictures. The pictures introduced here are not strong reinforcers and may depict verbs, actions and concepts. Finally when the sentence is constructed the communication partner will read the pictorial sentence aloud and the child will be encouraged to repeat the sequence, thus promoting oral communication.

There are many reasons for using PECS. The system is not complicated to adopt and it does not require long and complex training for the PECS trainees. Furthermore, other communication partners will easily be able to pick up the system without any training. No expensive equipment is needed, and the system can be applied in many different environments and situations. The materials required will be elaborated in the following.

3.3.2 Pictures and The PECS Book

Pictures are the core elements of PECS. In the early stages of the training, pictures illustrate desired items. Later on in the training program, more complex pictures are introduced. Text may also be displayed along with the pictures. It is not required that the child can read in order for text to be displayed along with a picture, as the child will learn to associate the text with the item. The presentation of the pictures can be tailored to an individual child's learning style. Size, colour and even 3D effects can be modified to encourage the child's communication development. Both the text and the picture itself can be altered in size to promote learning. Pictures can symbolise anything such as; items, actions, complex terms or even have a special meaning [BF94a]. Actions are usually composed of verbs and complex terms will illustrate abstract phenomenons as seen in Figure 3.1. An example of pictures with special meaning, is the "Help me" picture. The child has been taught that if it needs help to use this picture as an exchange. Underneath examples of all four categories of such pictures can be seen in Figure 3.1.



Example of a desired item (Grapes) | Example of an action (Eat) | Example of special meaning (Help Me) | Example of a complex term (One-to-one) & (Unassisted)

Figure 3.1: Examples of PECS pictures photographed at the first collaborative meeting.

With the increase in number of pictures introduced throughout the training of PECS

a place for the child to organise and store their pictures is necessary. Such a place often used is a dedicated PECS book. A PECS book provides the ability to sort the pictures into pages. Pictures that the child uses most frequently may be placed on the front page, and the communication partner can rearrange pictures to suit the given situation. When the child wants to request a particular item it can place the corresponding picture onto a sentence strip that is located on the cover of the book. On this strip the child can also construct sentences, that can be passed to communication partners with the intent of initiating communication. A child should only have a personalised book which must be kept in a place easily accessible [BF99]. An example of a PECS book with accompanying sentence strip and pictures can be seen in Figure 3.2.

Figure 3.2: Example of a PECS book photographed at the first collaborative meeting.

3.4 Problem Definition

In the introduction to this chapter we stated the desire to work with autism and PECS. The understanding gained throughout this chapter allows us to transform this desire into a more specific problem statement. We defined that we would like to digitise PECS. PECS however is a comprehensive system. Children are taught the system in six phases and they use a broad variety of tools. We believe that making a digital solution covering all of these aspects would introduce some problems. Especially in the first phases of teaching PECS, much effort is spent on teaching the child the concepts of physically handing over distinct pictures in exchange for desired objects. We believe this concept is hard to emulate using a tablet, or phone device, which has to contain and represent all these pictures in one single device.

Instead we would like to focus on digitising their most valued tool – the PECS book. This leads us to the following problem statement:

> *How can we design and develop a digitalised solution of the PECS book, which is to run within GIRAF, used to assist children with autism, in communicating with their communication partner?*

This digital solution will be an application, called DigiPECS, running in GIRAF and available through GirafPlace. DigiPECS should contain two modules. A module allowing guardians to maintain the PECS book through the central administration interface offered by GIRAF and a module for allowing the children to interact with the book. Further, knowledge and understanding are gained through our collaboration partner Birken. A meeting with this partner is presented in Chapter 4. The results from this meeting will be analysed and used to specify the requirements to DigiPECS. This is accomplished in Chapter 5.

DigiPECS will be developed using an agile development method inspired by Scrum. This development method will be presented in Chapter 6.

CHAPTER 4

COLLABORATION PARTNER BIRKEN

To be able to digitise the PECS book in a way that can aid children with autism, it is important to understand the special needs they have. Parts of this research can be accomplished through reading, but to get a profound understanding of the way children with autism communicate with a communication partner, we need to talk to both. Further, observing how communication between the child and the communication parter takes places, will give a deeper understanding of their needs.

To get this understanding, we have contact to a daycare centre, *Birken*. Birken specialises in children at the age 0-7, diagnosed with ASD (for further explanation, see Section 3.2) or other closely related mental development disorders, [Kom10]. Birken can accommodate 30 children divided between two locations in Vodskov and Langholt. The employees at Birken are all pedagogues and further specialists within physiotherapy, occupational therapy and speech/hearing consultants are affiliated.

The primary focus of Birken is to give the children a safe everyday life, as well as teach the children social skills and thereby making family life easier. The effort is put on the development of the child just like the well-being of the child is a stated objective. To aid these objectives e.g. PECS is used to enable a child, with no or limited spoken language, to communicate. Further, computers are used as an assistive technology to train social interaction.

Bente Thrysøe has attended a course in PECS, and is associated to Birken. We have set up two meetings with her at Birken. First meeting is to establish the understanding needed to develop DigiPECS mainly comprising how PECS is used today. This initial meeting took place during pre-game. The output of the meeting is presented later in this chapter. Second meeting took place after development with the purpose to perform a usability test, using the children. This is elaborated on in chapter 11, where an evaluation of the system takes place.

4.1 Method

The main objective of the first meeting with Birken is to understand how PECS is used in the analogue form. This way, we will get a basis knowledge of whether a need for a digitised PECS exists.

To prepare for the meeting, a list of questions regarding the PECS book and usage of it was made. Also, a prototype, based on reading about autism and PECS was produced. The prototype, as well as changes made to it, is presented in Section 10.1.1. The prototype was made to present the initial idea, but also to, through feedback from Bente, gain insight in the children's capabilities and limitations.

4.2 The Meeting

The meeting in Birken was divided into two phases. First phase we had the chance to observe a *breakfast situation*. Sitting at the table with the children and the pedagogues, we saw how the communication through the PECS book took place. After breakfast a meeting with Bente Thrysøe was scheduled, and giving us the opportunity to present the prototype and ask questions to clarify the usage of the PECS book. In the following two sections, a summary of the observations derived from the meeting is presented.

4.2.1 A Breakfast Session

The following observations were found during breakfast:

- Children at the table had each their individual PECS book customised their needs by their side.
- Some children had many pictures to choose from (milk, water, bread, crispbread, butter, banana, apple etc.) while others only had two (drink, food).
- Some children placed the picture from the book on the sentence strip to communicate, while others handed the picture directly to the pedagogue. Some children needed help to take the picture.
- Whenever a child communicated through PECS, the pedagogue reads aloud the word on the picture. The child tried in accordance with its communication skills to repeat the word.
- Every item in front of the child (plate, glass, knife) had a fixed place painted on a dinner mat.
- A short 'schedule' of the breakfast situation was placed beside each child showing how the breakfast should happen. Ending with a picture indicating 'Tak for mad' (Thanks for food).
- If a child finished eating before permitted to leave the table, it was given an hour glass, showing when the child may leave the table.

4.2.2 A Meeting With Bente

During the meeting with Bente Thrysøe, the following facts were found:

- Children with autism are as interested in computers as other children, and have technical skills at the same level.
- Each distinct situation has its own PECS book. E.g. eating, 'choose-book' (Danish: Vælgebogen - represents the toys available)
- Representation of text and image on a picture is adjusted to the child. The more experienced the child has with the PECS book, the daycare center makes the images smaller and the texts larger. This is to improve their skills in writing and constructing sentences.
- If a child places pictures in a wrong sequence on the sentence strip, the pedagogue moves pictures in the correct order and read the sentence loud.
- If an item is not available for the child to choose, the picture representing it is either moved to a red page in the PECS book indicating it is unavailable or it is removed from the book.

- There are not enough space in the PECS book to all the pictures some children need.
- A picture representing 'I want' is placed on the sentence strip from the beginning, preparing the children to construct sentences.
- Pictures in the PECS book are individually adjusted to the child. Some children like concrete pictures of an object - taken with a camera - while others like neutral pictograms with no colours.
- The same pictures are used in different contexts, e.g. day planning, story-telling etc.
- Verbs are typically placed at the left side in the book, starting from the top and down.
- 'Help me' is a special picture with two hands reaching out for each other.
- Some children like to organise their own pictures according to their own system
- When adding a new picture to the PECS book, it is made on the computer, printed and laminated, and velcro is placed on the back of the picture.
- Some of the children have a repetitive behaviour and do not like unexpected events. The system should take this into account.

4.3 Conclusion

The knowledge obtained through the meeting in Birken aided in understanding both opportunities as well as challenges in a digitised PECS. Later in the report, cf. Section 5.2, the functionality to be implemented will be defined, but to promote a justification of why PECS could benefit from being digitised, a list of advantages and challenges are summarised below. The reasoning for the following list should be found in the meeting with Birken as well as reading material studied related to PECS and autism.

- Adding pictures to the analogue PECS book requires that a picture is already available or that one has to be created. Creation can be done by drawing either in hand or on a computer, or by taking a photo and print it out. By digitising PECS adding a picture can be done easily. The mobile platform has an integrated camera. A picture taken with the camera is available for use in the PECS book immediately.
- In the analogue form of PECS, a child with no oral communication skills, has to address a communication partner to hear a word associated with a picture pronounced. A digital version of PECS could offer the function of read aloud the words/pictures that are chosen from the book. Further, the whole sentence could be read aloud as many times as requested. This could potentially enhance the oral communication skills of the child.
- Physical pictures – even when laminated – will over time, if used on a daily basis, become worn and have to be replaced. This is avoided in a digital version.
- The way the PECS book works today, a picture has to be manually removed/inserted from/into the book, when the option is available for the child. E.g. if the same book is used both in the daycare and at home, some pictures have to be present one place but not the other. In a digitised PECS this could be automated. A setting for the communication partner could be to enable or disable pictures, making it possible to show breakfast-options from 8 am. to 9 am., daycare-options from 9 am. to 3 pm. etc. This could help to structure and control which options should be available when.
- Pictures can get lost - dropped or removed by the communication partner. Lost pictures can be avoided if they are stored on the mobile unit.

- By digitising PECS all pictures will be in one place which will ease the mobility.
- Based on the meeting with Bente Thrysøe from Birken, a common characteristic of a child with autism (as well as most other children) is the fascination of electronics. They are very fond of computer games, and have no problem with navigating in the virtual world. Introducing the children to a digitised PECS could perhaps make the teaching of communication a game and thereby enhance the learning outcome.

Though more aspects of PECS can obtain a possible benefit from being digitised, some challenges are also introduced. Some can be overcome during implementation by considering the user-group of the system along with any special needs identified. But other challenges – like the robustness of the platform – is a matter that speak against a digitisation.

- A mobile unit, e.g. a tablet, is fragile. If it is dropped on the floor, will it stand the impact?
- A mobile unit is costly and fragile. So if the child is going to the playground or the beach, will the unit be brought?
- Learning to communicate through PECS will in the early stages only require a few pictures that can all be visible to the child at the same time. But an increase in numbers of pictures will rapidly require to expand over more virtual pages. For the child to be able to follow the concept of pages on the unit, the digitised PECS has to be intuitive to use.

CHAPTER 5
SCOPE OF PROJECT

In Chapter 3 and Chapter 4 the focus was to obtain knowledge about autism and PECS through investigation of these areas and a collaboration meeting with the day care centre Birken. In this chapter, this knowledge is used to answer two main questions. Who will be interacting with DigiPECS and what information should DigiPECS handle?

These questions will be answered in two sections. In Section 5.1 the application domain will be determined by identifying and describing the users of DigiPECS. In Section 5.2 the problem domain will be determined by identifying key objects which have to be handled by DigiPECS. The result of this chapter is a system definition presented in Section 5.3. The section will present a FACTOR for DigiPECS together with a short delimitation. This will define the scope of DigiPECS and will be used as a reference throughout the development process.

5.1 Application Domain

From Chapter 3 and Chapter 4 several users with interest in PECS were informally introduced. The child, just diagnosed with autism, children using PECS at an early level and children with extensive experience with PECS. There is also the parent using the system at home, the pedagogue who uses the system for educational purposes or maybe the grand parents using the system occasionally when the grandchild is visiting. Even though these users have slightly different backgrounds and goals when using the system. It is possible to aggregate these into user roles. In [Coh04] a user role is defined as:

> A user role is a collection of defining attributes that characterize a population of users and their intended interactions with the system

Using this definition, two distinct user roles emerges. The first role is related to users with limited capability to communicate vocally. These users utilise PECS to construct communication messages. The other role is the assistant, in Section 3.3 defined as the communication partner, handling and responding to these messages. These roles relates well to the multi-project GIRAF, which defines two types of users. A child using the system and a guardian assisting the child when needed. In DigiPECS the names of these roles are inherited from GIRAF. In the following, each of these roles will be described:

Child The users in this group all have a form of Autism Spectrum Disorders (ASD), introduced in section 3.2, resulting in the lack of capabilities to perform verbal communication. To help them communicate they have been introduced to PECS. PECS is taught in six phases. But as seen in the problem statement, Section 3.4, the focus in this project is

to develop a digital version of the PECS book and not to develop a PECS learning tool. In this project we choose to span all PECS users into a single role even though their capabilities to use PECS varies. This variation in capabilities was obvious from the collaboration meeting. The amount of pictures, the size and shape of these and their organisation are all adjusted to suit the individual child's needs. These capabilities do not necessarily depend on their age or the amount of time they have used PECS, but more often on the degree of autism the child experiences. From this meeting we also learned that these users prefer a predictable environment and a structured daily life. Relaxation for these children is having a well defined task to perform while too many choices, and too much freedom, brings confusion and discomfort. Children with autism can be taught PECS as early as two years old, but it is often used by much older children as well. One of the goals with PECS is to teach the children to communicate vocally but not all children achieve this goal, meaning the age of these users can span widely.

Guardian In general, a guardian is a person who looks after and is legally responsible for someone who is unable to manage their own affairs. In case of DigiPECS this is the communication partner, responsible for receiving, and responding to, communication requests from the child. From the collaboration meeting we learned that a large amount of these guardians' time are spend on creating or modifying pictures for PECS or to reorganise the contents of PECS books to suit a given situation. They also assist the children constructing sentences or rearrange these in case the sentences are constructed wrongly. PECS is however not only used by pedagogues. PECS is often used by parents to assist communication at home. Here a series of pictures can be used to express a trip or a task. PECS can also be used by other family members when visited by the child. Commonly among these users are the behavioural patterns, adding and manipulating pictures, receiving sentences and so on, are basically the same. This is why we choose not to distinguish between these users.

5.2 Problem Domain

With the users defined above, this section focuses on the problem domain. The problem domain defines the kind of information the system should handle. From the experience gained in the initial collaboration meeting we identify the objects that are important to the user and have to be controlled by DigiPECS. Because the system should be easy to learn and use for existing PECS users, the concepts of these objects should be recognisable in DigiPECS. For each object, a short definition is presented together with their relation to DigiPECS. At last functionality requirements, concerning these objects, are presented.

PECS picture A PECS picture is a graphical representation of an item, like a toy or a description of an action. This picture is, individually or in a sentence, exchanged during a communication session. In DigiPECS this notion is kept but in the process of digitising this object, the following functionality should be supported:

- Creating a new picture in the analogue PECS book is a work intensive process. In DigiPECS it should be possible to add new pictures either from the device storage or directly using the integrated camera. This should allow for a more personalised PECS book which can easily be updated to reflect changes in the child's environment.
- It should be possible to add text to an image and edit it afterwards. It should also be possible to vary font and size.
- One of the advantages of a tablet device is its capability to both play and capture sound. In DigiPECS it should be possible to add sound to a picture or edit it afterwords. This sound should be played back when the user interacts with the picture.

This repetition of a pictures pronunciation or meaning should help the children develop their capabilities to understand and express speech.

Sentence In PECS a sentence is a series of pictures describing a desire or request. Each picture represents one or more words and this use of multiple pictures allows construction of more complicated communication messages. In DigiPECS a child should be allowed to create these sentences by moving pictures from the book to a sentence-board. The functionality requirements are the following:

- It should be possible to create sentences by moving pictures into the sentence board. Through our initial collaboration meeting this feature was evaluated and found to be a usable solution. The focus is both to make it fast, intuitive and fun to use and to increase the available screen space for pictures by avoiding use of unnecessary buttons.
- It should be possible to play out the pronunciation of a whole sentence. The goal with this feature is to enhance the child's capabilities to recognise and understand sounds and, in the longer run, to develop some oral capabilities.

Sentence Board In PECS the sentence strip is the area in the book where pictures can be placed to form sentences. The name is due to the fact that this strip can be taken off and given to a communication partner. In DigiPECS the concept will remain but this area will be called a sentence board instead. The functionality requirements are the following:

- It should be possible to rearrange pictures in the sentence board or remove them completely. As described in Section 4.2.2 children are often creating sentences incorrectly, and for educational purposes, are receiving guidance in rearranging these.
- As stated in Section 4.2.2 pictures describing verbs like "I want" are often placed permanently in the sentence strip. DigiPECS should support this behaviour by allowing one or more pictures to be locked in the sentence board.
- The whole sentence board should be visible even though the child is in an early PECS learning state and only uses sentences consisting of a single picture. This requirement is here to ensure they grow accustomed to the existence of this sentence board.

Situation In DigiPECS a situation will be a collection of pictures related to a given context. In Section 4.3 a common procedure was identified. Often a guardian is reorganising the content of a PECS book with the desire to limit the number of available pictures for the child to suit a given context. As stated earlier a digital solution would be able to make this task easier by allowing a guardian to save and load different sets of pictures. The functionality requirements are the following:

- It should be possible to create, modify and delete situations using the tablet device.
- It should be possible for a guardian to select one or more situations and lock these to the pecs book.
- It should be possible to load situations automatically according to the device position or time. The goal with this feature is to allow a parent to create specific situations with toys or other objects which is only available to a child on a single place or in a time frame.

Categories As realised in the initial collaboration meeting, the amount of PECS pictures used in a kindergarten is high. These pictures are organized in different pages in a single book or even in several books. This large amount of pictures need organisation in an archive, such that guardians may quickly find the desired picture. The organisation can be accomplished by categorising the pictures by similarities, such as food items, drinks, and activities.

5.3 System Definition

As with GIRAF, the initial agreement and overall requirements of DigiPECS is stated by use of a system definition. Throughout the development, this system definition should be referenced to ensure the stated prerequisites are met. In the following, the FACTOR for DigiPECS is presented:

Functionality DigiPECS should offer functionality to:

- Allow a child to construct sentences consisting of pictures selected from a library.
- Allow a guardian to create and organise pictures in the library, possible in categories.
- Allow a guardian to limit the functionality according to the child's abilities or situation.

Application domain DigiPECS is targeted at children with autism and their guardians.

Conditions The children are assumed to have been diagnosed with autism. Both children and their guardians might have limited IT-knowledge. Emphasis on usability is considered important.

Technology DigiPECS should primarily run on touch based units running Android OS 2.2.

Objects The objects identified in DigiPECS are: *Picture*, *Sentence*, *Sentence board*, *Situation*, *Category*

Responsibilities DigiPECS is responsible for allowing communication between a child and a guardian using a touch unit and Picture Exchange Communication System (PECS).

Using the FACTOR criteria defined above, we construct the system definition as follows:

> *DigiPECS will be a mobile application that facilitates communication between an autistic child and a communication partner - typically the guardian of the child - by digitising the ideas in the Picture Exchange Communication System (PECS). The application should offer the possibility of constructing sentences consisting of one or more pictures from a library.*
>
> *Further, the guardian should be able to administrate how and when pictures are presented for the child. This includes arranging the pictures into user defined categories.*
>
> *As the primary market segment is autistic children and their guardians, who can not be expected to have a profound IT-knowledge, the application must have a high degree of usability. The application should be developed for touch units running Android OS 2.2.*

5.3.1 Delimitation

Having set the initial requirements to DigiPECS, we need to define the scope of the project and to which extend PECS will be implemented.

- PECS is taught in six distinct phases and used to teach a child with autism how to communicate through the use of PECS pictures. The purpose of DigiPECS is *not* to teach how to communicate through PECS.
- GIRAF is targeted at *children with limited mental capabilities due to handicap or age*. DigiPECS is solely targeted at a subset of these children, namely children with autism having difficulties with oral communication and already familiar with PECS.
- As DigiPECS is representing a physical PECS book, the size of the hardware it is running on has to be somewhat close to the size of the physical book. Consequently, DigiPECS is primarily targeted the Samsung Galaxy Tablet with a screen size of 7 inches, making it possible to keep the representation of the size, as well as the number, of pictures on a page close to the physical PECS book. Still, it should be possible to adapt size and number to fit a smaller unit.

The consequences of these constraints can be seen in the end product in two ways. DigiPECS will not be optimal to use when teaching PECS, as a mapping of the first phase of teaching PECS is hard to digitise without loosing the interaction with the child as PECS is intended to give. Therefore, DigiPECS should only be utilised for children already taught how to communicate through PECS using the sentence strip. Further, by running DigiPECS on a smaller unit than an 7 inches screen, the emphasis put on a high degree of usability of DigiPECS must be reckoned to be lost, as getting an overview of the functionalities implemented is harder.

Part III

Development

CHAPTER 6

DEVELOPMENT STRATEGY

In order to keep the development of DigiPECS under control and on track, a development method has been chosen. The development method of DigiPECS is mainly inspired by Scrum. DigiPECS is a combination of a predictive and inventive project. DigiPECS is predictive due to the fact that it is a digitisation of an existing communication system, but inventive as we have to find a way to map the physical book into a digitised book without loosing the essence of the book.

Scrum has been solely used for inspiration for our method, that means we have adapted it to fit our needs. Here, a short and generally introduction to Scrum will be presented, based on a book and an article[Sch97] [Lar03]. Afterwards we will describe our method in detail and lastly state the reasons for our choices.

To characterise a project to be developed using Scrum, the following 'rules' have to be followed accurately. Scrum, in its original form, consists of four phases, *planning*, *staging*, *development* and *release*. *Planning* in Scrum aims at stating the vision of the product as well as setting up an initial *Product backlog*, consisting of requirements to implement, and estimation of costs. Further, prototyping and experimental designs are created. *Staging* is the phase where further requirements are identified and included within the product backlog. The phase ends with planning the first iteration, by constructing a *Sprint backlog*. The sprint backlog contains a list of features to be implemented during the sprint. Combined, Planning and Staging are called Pre-Game.

Development is implementing functionality in iterations called sprints. Each sprint in Scrum typically has a duration of 30 days. During a sprint it is not allowed to add new features to the sprint backlog - features should only be added at the start of a sprint during the *sprint planning meeting*. Every morning a Scrum meeting is held – *daily scrum meeting* – in order to clarify what is *done*, what will *happen* and are there any problems - *blocks* - standing in the way. *Release* is the phase where preparations for release such as documentation, testing and integration takes place.

Scrum makes use of different roles. The *Scrum team* is a group of developers, working in a sprint. *Product Owner* is responsible for creating and prioritising the product backlog. *Scrum Master* is responsible for the sprint backlog and conducts the daily scrum meetings.

6.1 Development Method for DigiPECS

As mentioned, the development method of DigiPECS is mainly inspired by Scrum. The process of development has been divided into two phases. *First phase* for gathering knowledge and *Second phase* for the development process and implementation. The overall development method is shown in figure 6.1.

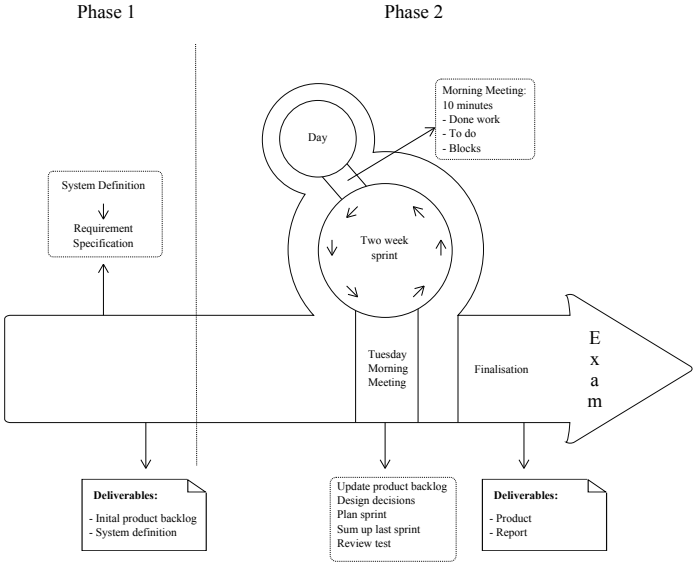


Figure 6.1: An overview of the development process for DigiPECS

6.1.1 Phase One

First phase, the *pre-game*, encompasses a preliminary analysis with the objective to obtain a preliminary product backlog. Inspired by Object Oriented Analysis & Design (OOAD), we have made a preliminary analysis concluding with a system definition which serves as a common understanding and agreement for the group to start the further development. We investigated the existing physical PECS as an exemplar. Besides this, we made some paper prototypes to understand how the flow through the DigiPECS could operate. With the system definition, the preliminary product backlog was created, consisting of user stories stating *who* wants a *functionality* in order to obtain a *goal* (cf. Section 8.2). Further, it was planned how the rest of the development should take place. It was decided that each sprint (iteration) for the rest of the project should last *two weeks* each, starting Tuesday morning with a planning meeting with the purpose of planning next sprint and review test.

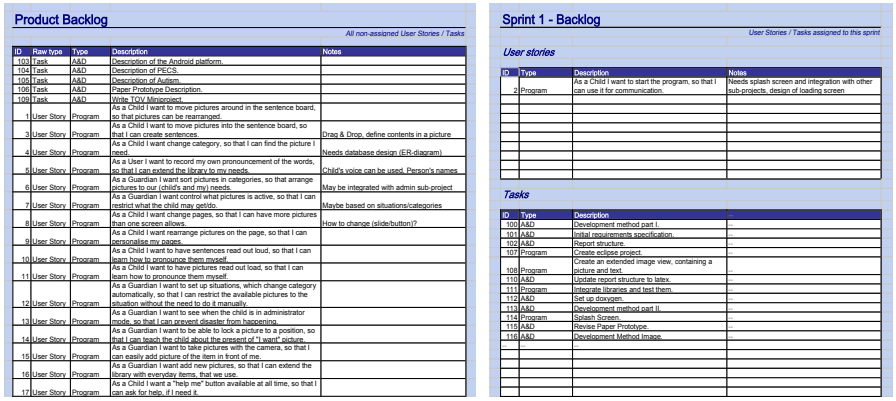
6.1.2 Phase Two

Second phase of the development is the implementation of DigiPECS, corresponding to the development and release phases from Scrum. The main tools used for planning and managing the project are *product backlog, sprint backlog* and *sprints*. Further, inspired by Extreme Programming, *pair programming* has been used through parts of the implementation.

To support the planning of the project, both an online product backlog and an online sprint backlog has been created, to assure that all members of the group have constant access to the work process. An example of a sprint backlog with its a appertaining *user stories* and *tasks* along with the interim product backlog (as was on 25th March), is shown in figure 6.2.

Secondly, a wall in the group room is dedicated to get a visual overview of how much progress is made in a sprint. A post-it represents a task, and it is placed on the wall in categories according to its state (Planned, Started, Testing and Done), see figure 6.3.

A timeline showing the landmark events of the overall development process of the project is depicted in figure 6.4. It is to a great extend inspired by the Scrum development



Product Backlog

All non-assigned User Stories / Tasks

ID	Raw type	Type	Description	Notes
103	Task	A&D	Description of the Android platform.	
104	Task	A&D	Description of PECS.	
105	Task	A&D	Description of Autism.	
106	Task	A&D	Paper Prototype Description.	
109	Task	A&D	Write TOV Miniproject.	
1	User Story	Program	As a Child I want to move pictures around in the sentence board, so that pictures can be rearranged.	
3	User Story	Program	As a Child I want to move pictures into the sentence board, so that I can create sentences.	Drag & Drop, define contents in a picture
4	User Story	Program	As a Child I want change category, so that I can find the picture I need.	Needs database design (ER-diagram)
5	User Story	Program	As a User I want to record my own pronouncement of the words, so that I can extend the library to my needs.	Child's voice can be used. Person's names
6	User Story	Program	As a Guardian I want sort pictures in categories, so that arrange pictures to our (child's and my) needs.	May be integrated with admin sub-project
7	User Story	Program	As a Guardian I want control what pictures is active, so that I can restrict what the child may get/do.	Maybe based on situations/categories
8	User Story	Program	As a Child I want change pages, so that I can have more pictures than one screen allows.	How to change (slide/button)?
9	User Story	Program	As a Child I want rearrange pictures on the page, so that I can personalise my pages.	
10	User Story	Program	As a Child I want to have sentences read out loud, so that I can learn how to pronounce them myself.	
11	User Story	Program	As a Child I want to have pictures read out load, so that I can learn how to pronounce them myself.	
12	User Story	Program	As a Guardian I want to set up situations, which change category automatically, so that I can restrict the available pictures to the situation without the need to do it manually.	
13	User Story	Program	As a Guardian I want to see when the child is in administrator mode, so that I can prevent disaster from happening.	
14	User Story	Program	As a Guardian I want to be able to lock a picture to a position, so that I can teach the child about the present of "I want" picture.	
15	User Story	Program	As a Guardian I want to take pictures with the camera, so that I can easily add picture of the item in front of me.	
16	User Story	Program	As a Guardian I want add new pictures, so that I can extend the library with everyday items, that we use.	
17	User Story	Program	As a Child I want a "help me" button available at all time, so that I can ask for help, if I need it.	

Sprint 1 - Backlog

User Stories / Tasks assigned to this sprint

User stories

ID	Type	Description	Notes
2	Program	As a Child I want to start the program, so that I can use it for communication.	Needs splash screen and integration with other sub-projects, design of loading screen

Tasks

ID	Type	Description	...
100	A&D	Development method part I.	..
101	A&D	Initial requirements specification.	..
102	A&D	Report structure.	..
107	Program	Create eclipse project.	..
108	Program	Create an extended image view, containing a picture and text.	..
110	A&D	Update report structure to latex.	..
111	Program	Integrate libraries and test them.	..
112	A&D	Set up doxygen.	..
113	A&D	Development method part II.	..
114	Program	Splash Screen.	..
115	A&D	Revise Paper Prototype.	..
116	A&D	Development Method Image.	..

Figure 6.2: An extract of the product backlog to the left, an example of a sprint backlog to the right

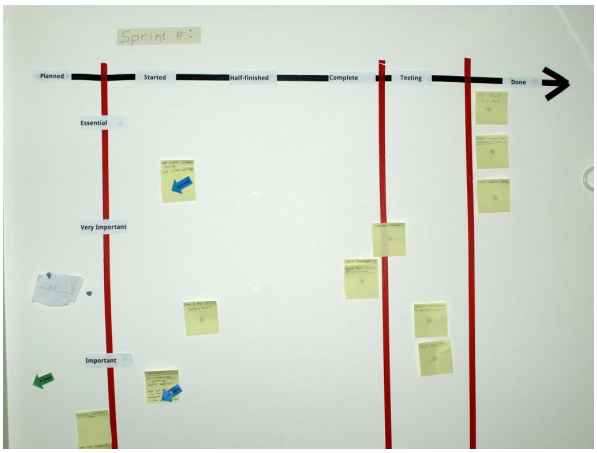


Figure 6.3: Visual overview of the process

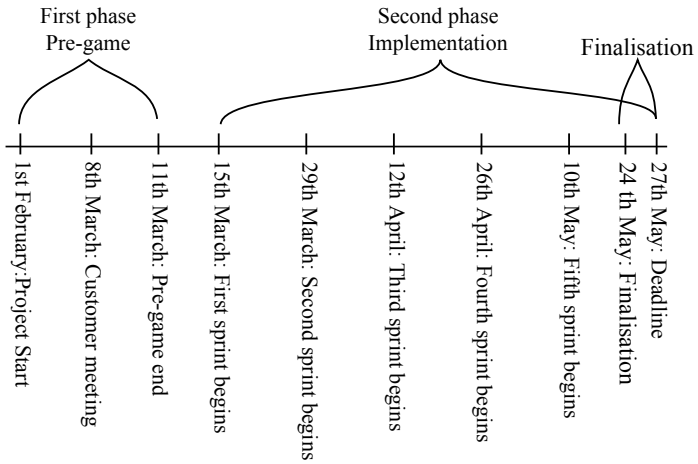


Figure 6.4: A timeline showing the landmark events of the project

method and shows the development from planning to release.

6.1.3 Reasoning for Development Method

As this project has a fixed delivery deadline, it was important to have a timely aspect. A traditional development approach, such as a classic waterfall model, typically places all analysis and design up front, and implementation and testing will be the last part of the project. This would for the development of DigiPECS not be optimal, as this could mean a risk of "running out of time". Eliminating implementation and testing of functionality, which already has been thoroughly analysed and designed, could be consequences.

Through the use of an agile development process model – in this case mainly Scrum – we can keep each sprint, as well as the whole project, time-boxed, and make sure that the deadline is not missed. The consequences of running out of time, using an agile approach, would similary also be lack of functionality in the finished product. However, all aspects of those features (analysis, design, implementation, and testing) is cut out of the project, entailing that no unnecessary analysis and design is composed.

It is important to prioritise which features to implement in each sprint, and thereby ensure that we at the end of the project have a product encompassing the most important features. This ensures that only the least important features are cut out.

We did not, from the beginning, have all the requirements for the system, because we first had to understand what PECS is and how it is used at present. An agile development method is adaptive, which means that if new requirements are raised during the project, the requirements may easily be integrated into the implementation. A Waterfall method would require the whole analysis to be re-done.

The system is developed as a multi-project across four groups. This means that our group solely develop a part of the system, and do from the beginning not know how the rest of the system will be designed (which features are provided by others and which are not). Consequently, a thorough understanding of the system cannot be exposed in the beginning of the project. Using an adaptive method allows us to change and refactor our project, reflecting the changes made by other groups.

Moreover, an agile approach, is suitable for small projects as it depends on close communication in the development team with less documentation than in a large project where direct communication across all members can not be achieved. As DigiPECS is developed within a single group this is optimal for this project as the physical settings already exists.

Though the development of DigiPECS mainly is inspired by the agile paradigm, some modifications from the agile spirit was needed in order for it to suit this specific project. During this development there has not been an on-site customer present, who continuous may state requirements. Instead, during the development, only one meeting with the customer was planned during the pre-game phase. Therefore, the group had to "play" the role of a customer during the main part of the project.

Various initiatives are taken to ensure high quality code and an end system with a high degree of usability. Coding of complex and essential features is performed using pair programming, ensuring that all lines of code as been reviewed by at least one programmer. Further, different types of testing are applied. Clarification of test planning and execution is presented in Chapter 11.

CHAPTER 7

ANDROID PLATFORM & UNITS

Android is an open source operating system developed for mobile devices including phones and tablets. The maintenance and development are handled by the Open Handset Alliance with Google as the main contributor[Día11].

Usually mobile operating systems have been proprietary property. These operating systems are often targeting devices from a specific manufacturer or requiring other manufactures to obtain licenses to use them. Android is however preferring the Apache Software License 2.0[Inc11d] and is therefore not only free to use, it is also open source. The popularity has been steadily growing [Com11b] since its announcement in 2007[Rub07]. HTC, Motorola and Samsung are being the most popular manufacturers of Android Smartphones in U.S[Com11a].

Throughout this chapter we shall look at Android from both a hardware and a software perspective. This introduction is given, because it is important to understand the differences between developing normal desktop programs and developing applications for smartphones. Smartphones are typically smaller than laptops, and most of them are handled by using touch input instead of using a mouse.

Their small size makes the physical layout of buttons different. Resources like processing power, available memory and battery capacity are also limited. We will get to see the specific details which are important to understand, when designing an application for Android.

The hardware introduction will focus on the Samsung Galaxy Tab, a 7 inch tablet device, but also briefly introduce other devices. It will also include a small introduction to the Android Hardware Compatibility Definition. The second part will introduce Android from a developer's perspective and includes areas such as system and application architecture, including special considerations when developing applications.

7.1 Android Touch Units

Quickly looking at websites of Android Device manufacturers reveals a wide range of devices for a customer to choose from. The products range from smaller smartphones, like the Sony Xperia X10 mini (2.55 inch screen)[AB10] to larger tablets, like the upcoming Samsung Galaxy Tab 10.1v (10.1 inch screen)[SAM11]. There are also other devices than smartphones and tablets, which are using Android, however these are not considered relevant. Earlier, see Section 5.3.1, the Samsung Galaxy Tab, 7 inch edition, was chosen as the primary target platform for a DigiPECS application. This device will be presented in the following.

7.1.1 Samsung Galaxy Tab

At first glance, the Samsung Galaxy Tab may look as a heavily overgrown phone, which is rather impractical to carry within a pocket.

The Galaxy Tab features a SIM card slot and dialling application, which makes it fully capable of handling phone calls. The SIM card slot is only one of several connectivity options, also including USB, Bluetooth, and WiFi, which allows the user to be connected at all times.

The main difference from a smartphone, is the use of a 7 inches screen, with a WSVGA resolution of 1024x600. The larger screen allows easier navigation and overview of complex content including websites and maps. The screen also emphasises the use of the Galaxy Tab as e-Reader and GPS navigator, as the information may be read from a distance.

As indicated the Galaxy Tab ships with a GPS unit. It also ships with two Cameras, a front facing 1.3MP camera and a rear facing 3MP camera.

Figure 7.1: The Samsung Galaxy Tab

The Galaxy Tab has three hardware buttons, and four touch sensitive buttons:

Power button primary for turning on and off the phone.

Volume up and down buttons for adjust the sound volume.

Menu button for accessing options available on the current screen.

Home button for accessing the home screen.

Back button for accessing the previous screen.

Search button for accessing search features.

The menu, back, and search buttons behaviour depends heavily on application implementations.

7.1.2 Other Units

In this project, two other Android devices are at our disposal. The HTC Wildfire is an Android 2.1 based smartphone with a 3.2 inch screen (240x320 QVGA). In many areas this unit seems identical to the Galaxy Tab but it is significant smaller in size and contains less processing power and memory. It does feature the same array of buttons, though some of them are located physically different. One main difference is that the Wildfire does feature an additional Optical Trackball, used for navigation in the interface.

Figure 7.2: The two HTC smartphones. The Wildfire on the left, and the Desire on right.

The HTC Desire is an Android 2.2 based smartphone using a larger 3.7 inch screen (480x800 WVGA). It is very similar to the Wildfire, with the exceptions that the touch sensitive buttons are replaced with physical buttons, and that the phone contains the same amount of processing power and memory as the larger tablet.

A wide range of other android devices are available. They usually differ in screen size, input options, computational power and form factor but are very similar in the array of buttons used. The similarities are requirements, made to ensure some uniformity and compatibility among devices. These requirements are stated in the Android Compatibility Definition.

7.1.3 Android Hardware Compatibility

Here, we will have a look at the Android 2.2 Compatibility Definition with a focus on the part concerning Hardware Compatibility. This part includes a list of hardware features that all Android devices must support. The description below only features some of the more interesting aspects of this definition[Inc10].

First this definition lists the most common display configurations. All these screen configurations are classified in either low, medium, or high screen density group. This is to help application developers to supply the correct graphic sizes. If the correct graphic size is not supplied it will be resized, by Android.

The definition also requires that all Android devices include support for software keyboards and that at least one is provided as default. It is also permitted to provide a hardware keyboard of either qwerty or 12-key type. As seen on the HTC units, it also permits the devices to implement a non-touch navigation option. Note that both the hard keyboard and non-touch navigation options are optional.

It is also in the Android Compatibility Definition we find the answer for the very similar button layout on Android units. All devices must implement a Home, Menu, and Back button. It is recommended to use dedicated buttons, however they may be implemented through software instead. In all cases it is required that they are accessible at all times – that is, they are not blocked by applications in the display area. The Search buttons, seen

on all of our devices, is recommended, but not required. Further, device implementations are allowed to include optional Send and End buttons for phone calls.

Other noteworthy details are that all devices must support change of screen orientation between landscape and portrait mode and they all must have touchscreens. Further, at least one wireless high-speed data connection must be available. This could be a mobile network connection like HSPA or a normal wireless technology like 802.11b/g/n. At last, a rear facing camera is also required.

Finally Android 2.2 may be used on devices that do not include hardware for telephony.

7.2 Developing for the Android Platform

Now that we know the hardware at our disposal, we are going to look into how to actually design and program for the Android Platform. First we are going to have a quick look at the architecture of Android to determine where in the system application is executed, and what is laying beneath them.

The Android platform also defines a set of components which are typically used in applications. This includes components to create "windows", to communicate between applications, and how to start other "windows".

Finally, mobile units often have less memory and processing power to execute several applications simultaneously. Therefore we are going to look at how Android handles the life cycle of an application.

7.2.1 Android Architecture

As seen in Figure 7.3, the Android operating system contains five major components [Goo11]. Android uses a modified *Linux kernel* for core system service management like memory, process, and network. It also acts as an abstraction layer for the rest of the system handling communication with hardware and sensors. On top of this kernel Android contains some *libraries*, written in C/C++, which are used by other system components. These includes libraries for handling graphics and databases. The *Android Runtime* handles execution of applications. These applications are written in Java and executed on a modified Java virtual machine called Dalvik. Every application is executed in its own process, both for security reasons, and to ensure that a crash of a single application does not influence the stability of the whole system. The *application framework* provides the developer with many building blocks for creating applications. These includes frameworks for user interfaces, background services and interfaces to components handling the phone functionality and more. On top is the *applications*, both the native ones and the user-installed applications.

7.2.2 Application Architecture

As mentioned earlier, applications are primarily written in Java, but due to limited system resources on mobile devices, only a subset of the Java SE specification has been implemented in Android. An Android application consists of a set of components loosely coupled together. How these components are bound, and other meta data for the application, is specified in a manifest file. Some of the components that typically make up an Android application are [Mei10]:

Activities is used to present something to, or take input from, a user. An activity is equivalent to a form known from desktop development.

Service components is used to perform tasks whenever an application's activities are not visible. In a music player, a service might control the playback of music while the user interacts with other applications.

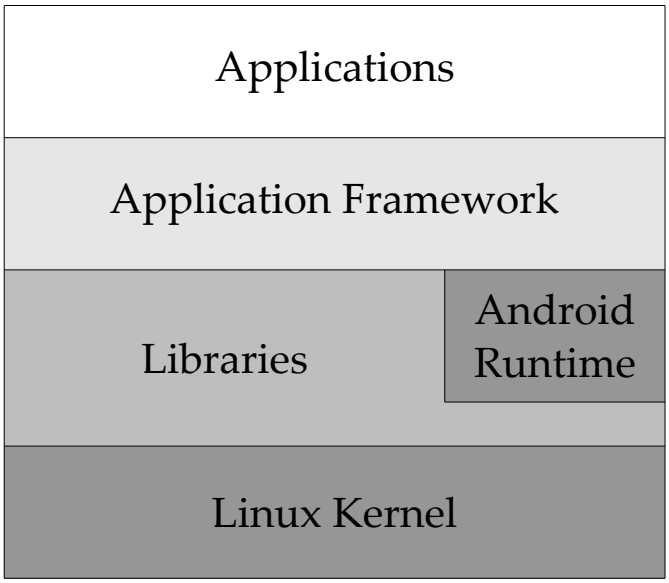


Figure 7.3: An overview of the android system architecture

Content Providers are used to manage and share databases between applications. Android contains several native content providers exposing contact details and media files.

Intents are a means of inter-application communication. Intents can be explicit, a message to an specific component, or implicit. An implicit intent is called a *broadcast intent* and specifies an intention. It is then up to the Android system to locate components capable of handling the intent. An example could be the user clicking on a hyperlink in an email. Android would then look for any application capable of handling the link and present the result to the user.

Broadcast Receivers are registered in an application to allow it to respond to different broadcast intents. A browser could register a broadcast receiver to start the application and show a website every time the user clicks a hyperlink in another application.

Activity Life Cycle

An Android application is allowed to remain in memory even though it is not currently being used. This ensures fast access to the most recently used applications. However, resources like memory and battery, are limited and Android will continuously kill applications to free up memory. This means an application's activities goes through a life cycle consisting of the following states: [Mei10]

Starts: An activity is started through the use of an intent.

Running: An activity is visable to the user, and is in focus.

Stopped: An activity has been closed correctly by the user.

Killed: An activity is killed, by Android, due to lack of memory.

When running low on memory, Android might ask an activity to finish or simply kill it. Every time a transition between states is made, methods are called. These methods can be overwritten by the developer to handle these state changes. For example by saving data

from memory to persistent storage when an activity is killed. The whole life cycle of an activity is presented in Figure 7.4.

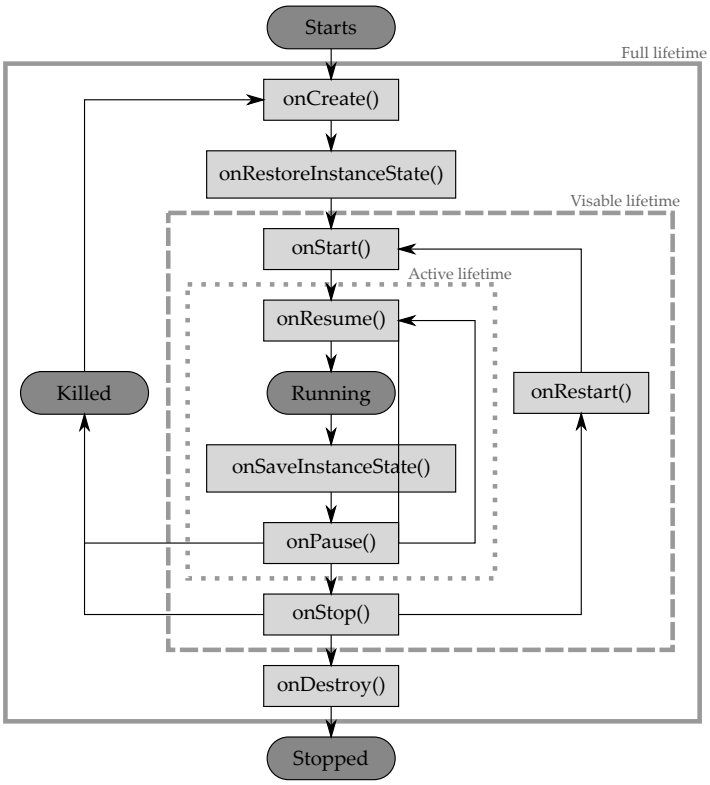


Figure 7.4: This figure shows the life cycle of an Android activity

Chapter 8
Analysis

The analysis phase of an agile software development process requires gathering enough information about the domain to understand the kind of system that is under development. This includes an understanding of what kind of system is about to be built and the reason of why the system should be built. This information is needed to aid the developers with prioritising features, which determines the construction order of the system. The phase is not executed in one single step as in the traditional development life cycle, but instead occurs throughout small adaptive incremental steps [Lar03]. This chapter will therefore be dedicated to documenting the final result of this phase in the development of DigiPECS.

A large portion of understanding the problem and application domain was identified in the pre-game (see Chapter 5). In this chapter we take a deeper look into how this information will be incorporated in the system. This includes finding and understanding the user requirements and how these relates to the development of DigiPECS, gaining a better understanding of the end users, and preparing non-functional requirements to the design and implementation of the overall system.

To help establish user requirements we will employ a common method used in both Scrum and XP. Here user requirements are primarily established by writing *user stories*, where a user story will capture functionality and features that are of value to the user of the system [Coh04]. Inspired by Scrum these user stories will in the development process of DigiPECS make up a Product Backlog, where a number of user stories in each iteration will be selected to be implemented. How to compose user stories optimally along with a short description of selective samples of stories from the DigiPECS Product Backlog with will be presented in Section 8.2.

To assist with writing user stories we need to understand the end users of the system. One way of accomplishing this task is to identify user roles. In the pre-game two user roles were identified Section 5.1, but it can be worth exploring these roles deeper. A well known technique is to make *persona* definitions. A broader description of what persona definitions are and how they fit into our development, is described in detail in Section 8.1.

As functionality is captured in the user stories there may still be a need to state some non-functional requirements to the system. This can be achieved by evaluating the overall quality of the system. To help define the quality of a system *quality attributes* may be applied. The definition of these and the rationale of each individual attribute are presented in Section 8.3

8.1 Persona Definitions

A persona is a fictive description, that is representative of a given user role. To help define a persona you can give the persona a name and describe specific details that will help bring the persona to life. The details should include behavioural patterns, background knowledge, goals and desires that are consistent with the user type you wish the persona to represent [Ben10]. Not every piece of information stated in a persona is necessarily of great importance, but should be intended to personalise the persona. Normally one to two persona definitions of each user role will be enough to represent the target group [Coh04].

In the development of DigiPECS we have employed the technique of creating persona definitions for three main reasons. First of all the persona definitions were employed to help us make the user stories more expressive. Secondly it will help us to better understand a user's situation when it comes to implementation of a specific user story. Thirdly we do not have a customer on site to help us through the design and implementation of DigiPECS. Due to these reasons we felt the need to further explore the target group, as the users are quite different than any user we have worked with before. In the pre-game we identified two types of users of DigiPECS: Children diagnosed with autism and their guardians. These are described in detail in Section 5.1. Additional details about these two types will be presented underneath in individual persona definitions.

8.1.1 Child

By studying the application domain both with literature and through the meeting with Birken we managed to gain a substantial amount of information about the children that use PECS and the appertaining PECS book. The following two persona definitions will describe common and interesting details discovered during this exploration.

Benjamin is six years old and was diagnosed with autism at the age of two. He is a very active boy, but has problems communicating verbally. He was first introduced to PECS in the daycare centre, when he was three. PECS gives Benjamin the opportunity to express desires and feelings through the exchange of PECS pictures. He has his own PECS book at the daycare and one at home which helps him organise his PECS pictures. Benjamin's PECS book contains multiple pages with at least fifty PECS pictures and as Benjamin is very attached to his PECS book he likes to organise the book himself.

Training with PECS has taught Benjamin to pronounce simple words such as "apple", "teddy" and "I want", and he is able to construct sentences with pictures illustrating complex terms. The illustrations on Benjamin's PECS pictures are getting smaller and the text more predominant, but Benjamin has had no trouble relating to the larger text. Because of Benjamin's autism, he prefers a predictable environment. Changes to his schedule or unforeseen events makes him stressed and uneasy. When one of his pictures disappears from his PECS book he gets notably upset and will not use the book until the picture has been re-added. Benjamin is strong visually and he likes to do large puzzles, but his favourite toy is his Nintendo ds. He finds it exciting and challenging, but he does not like it when the device behaves differently than expected.

Elissa was diagnosed with autism a year ago. She started attending a specialised daycare centre six months ago, when she turned four. Elissa has no verbal language and when she started at the centre she showed no signs of being able to communicate through any functional language. At the daycare centre, where a PECS consultant is attached, she has been introduced to PECS and can now communicate using pictures. Normally Elissa's training with the PECS occurs in repeated settings like at the breakfast table or the planned one hour playing time after lunch. Elissa has been quick to respond to PECS because she really likes that she with the system can ask for precisely the items she desires e.g. milk

at the breakfast table. During playtime she can even ask to play with her favourite toy by exchanging a picture of the doll with the pedagogue.

Elissa has all her PECS pictures in her own personal PECS book. She likes her book and knows precisely where to get it if she needs it. In her book she prefers actual photos of items as she can relate better to a photo of an object rather than an illustration. Elissa has between ten to fifteen PECS pictures in her book. When training with PECS begins, a pedagogue will help Elissa to organise her pictures, so the pictures that are relevant for the current setting are shown on the front page. A couple of weeks ago Elissa was introduced to constructing sentences of pictures on the sentence strip. She has not really understood how to construct a sentence yet, so a pedagogue has placed a "I want" picture in the sentence strip on Elissa's PECS book. Every time Elissa wishes to communicate she is encouraged to use the sentence strip and a pedagogue will help her arrange pictures on it. The pedagogue will also read aloud the whole sentence for Elissa when it is handed to her. Elissa's parents are contemplating using the system at home, as they can see that PECS gives Elissa the possibility to communicate.

8.1.2 Guardian

Two personas relating to the user role guardian is described here. They may have different intent of using PECS, but generally they both represent the user role called guardian.

Daycare centre pedagogue - Hanna

Hanna is 52 years old and is a pedagogue in Elissa's daycare centre. This daycare centre is specialised in working with children with autism. She is engaged in teaching the children communication skills using the PECS system. She likes this system because it is simple and intuitive to use and requires no specialised training on her behalf. She is no computer-wiz, but uses computers daily for reporting and to manage pictures used with the children.

In a workday Hanna has many working tasks. Every morning she prepares breakfast for the children. This involves organising the content of the children's PECS books to ensure only the relevant pictures for the breakfast session are shown on the front page of the PECS book. Special attention is given to the children that have trouble organising the pictures themselves. During the breakfast she receives strips of sentences or pictures from the children and responds by serving the desired objects and reading aloud the sentences or the text on the picture. The daycare centre has a broad collection of pictures that is digitalised, but she cannot always find the pictures that suits the individual child. Instead she often uses the digital camera and her computer to create new pictures for the children.

Parent - Peter

Peter is thirty-seven years old and father to a young girl Lisa with autism. Lisa has been introduced to PECS in her daycare centre and Peter have seen that PECS have given Lisa the opportunity to communicate. Peter thinks PECS has yielded amazing results relating to Lisa's communication skills and has adopted the system at home. He often creates his own PECS pictures, because he cannot always find illustrations that Lisa can associates an item with. He finds that using a camera to snap photos with, is the fastest and most convenient way of capturing items for Lisa's PECS book. It is mostly the family's digital camera that is used to take photos with, but more and more frequently Peter utilises the camera found in his new smart phone, purchased a couple of months ago. He is still learning the bit and bobs of functionality and services the phone provides, but loves his new gadget. Peter has an interest in electronics and gadgets, and tries to get the newest cutting edge devices when the family economy allows it. He is also introduced to exciting gimmicks through his work as design engineer at LEGO.

Peter and his wife spend an evening each month making PECS pictures for Lisa' PECS book. This includes printing out pictures, laminating these and applying them with at Velcro strip. Peter finds it a cumbersome task, and is looking for an alternative. As the family uses PECS many time throughout a day and using it in many situations, PECS is transported when needed. This means that PECS will accompany the family when it is time for supermarket shopping and family visits. This requires some organisation on Peters behalf, as it is quite important the pictures necessary for the upcoming trip are planned and packed.

8.2 User Stories

As described our requirements to DigiPECS are comprised in user stories that represent concrete features and functionality important to the user. This approach is common in agile development[Coh04], as the focus is to verbally communicate with the user or customer of the system rather than document an often technical upfront requirement specification.

A user story normally consists of three elements; *a written description* used throughout the project for planning and as a reminder of what has to implemented, *notes* that exhibit details about the story learnt through discussion both when writing, planning and implementing a story, and finally *tests* that will document details and determine if and when a story is completed [Coh04]. As a user story represents a customer's requirements. User stories should not state any technical details, but be written in a language that can be understood by all. Also it is beneficial to give the users a central role when formulating a user story. One way of accomplishing this is to write the story on the from of [Coh04];

"As a <*role*> I can <*functionality*> in order to obtain a <*goal*>"

We have adopted this approach as formulating user stories are new to us. In addition it has been of great importance to keep focus on the user of the system when developing DigiPECS. By constructing user stories this way, the specific user role, personalised with persona definition, will be easier to identify with when it comes to implementing specific user stories.

8.2.1 User Story Diagram

One way of writing user stories is to have a workshop where developers and the customer get together to write and discuss the initial user stories. In our case we held a group workshop between the members of the group after the first meeting with Birken. To conduct this workshop we took one user role at a time, and placed the starting user story in the centre of a black board. In case of the user role *child* the outcome of the starting user story involved starting DigiPECS, resulting in user story two (see Section 8.2.2). From this point on we drew stories from the user perspective extending the starting user story keeping in mind of what the user wanted to do now. The resulting diagrams of each user role can be seen in Appendix B and a small combined diagram can be seen in Figure 8.1. Further description of a selected number of user stories are presented in the following.

8.2.2 Detailed Description of Selected User Stories

A selected list of the user stories is presented here. With each story a description stating motivation, ideas and considerations discovered during the discussion, design and implementation of the distinct stories. This description is expressed as a result concluded at writing time. A complete list with user stories and user story test notes is available in Appendix A.

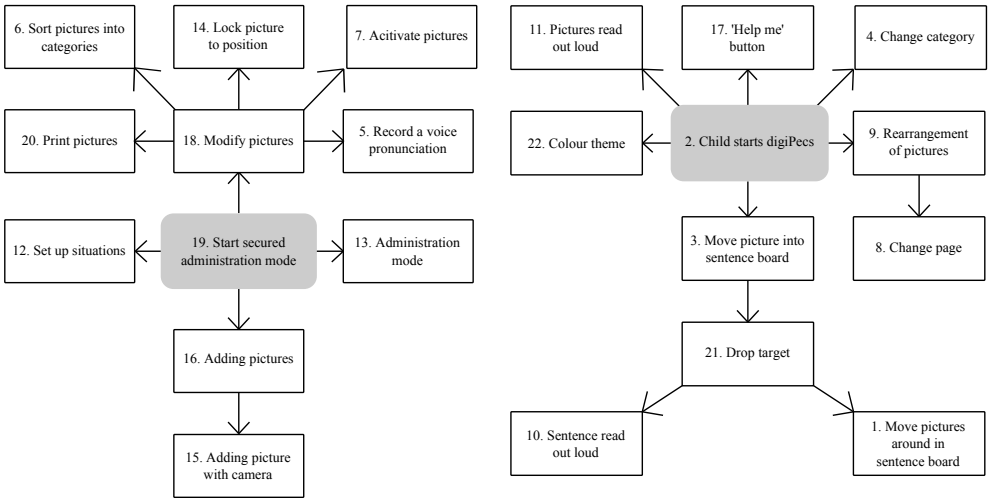


Figure 8.1: User Story Diagram for the User Roles **guardian** & **child**

2: Child starts DigiPECS : *As a child I want to start the program, so that I can use it for communication.* This is the entry point of the DigiPECS application. The goal is to bring the child from the GIRAF launcher to the DigiPECS book in a predictable environment. An easily recognisable icon is needed, so the child never will be in doubt where to find DigiPECS in the launcher. The focus should be on predictability and speed and not on options, as we imagine that the program is accessed many times throughout the day. Every time the program is started, it is important that the child's own personalised DigiPECS book is shown, with the same pictures as when the application was last exited. However, the front page should be shown every time the application is started. When the application is resumed the program should resume state, so nothing in the session is lost. This would mean that when a picture has been left in the sentence strip or a certain page has been accessed, this should also be visible when DigiPECS is resumed.

3: Move picture into sentence board : *As a child I want to move pictures into the sentence board, so that I can create sentences.* The main goal with this user story is to ensure a child can move pictures from a page to the sentence board, thereby training sentence construction with child. This can be achieved in multiple ways, but through the meeting with Birken it was determined that using drag and drop functionality is a feasible solution. Drag and drop, closely imitates the actual physical hand movement of taking a picture and move it into a sentence strip found in the analogue PECS book. To guide the child in the use of drag and drop an indication, either visible or audible, that the a picture has been dropped into the board may come as a help. When a picture is added to the sentence board it should also be possible to delete the picture from the board. What should happen when a picture is already situated in the position, have been discussed through the means of paper prototyping. It was decided that the newly added picture should override the picture already found in this position.

It is also quite important that the sentence board is visible at all times throughout the navigation of the application, as this resembles how the analogue PECS book works. This indicates that the sentence board should resume state when the application is running. However, it may give a better user experience to clear state the moment the DigiPECS is exited. A number of pictures has to be defined, as the number of pictures that can be dropped into the sentence board. Four appears to be a reasonable number, but this will depend on the size and density of the device screen.

8: Change page : *As a child I want to change pages, so that I can have more pictures than one screen allows.* Contrary the physical PECS book, DigiPECS can include an almost arbitrary number of pictures suggesting it should be possible to organise pictures. A few options are available such as horizontal, vertical scroll, but we have opted to organise pictures into pages, so DigiPECS closely resembles how this is accomplished in the analogue PECS book. A fixed numbers of pictures should be available on each page.

For a child to know which page that is currently in focus, the interface will need to indicate the current page number. A page number line could visualise this. It is important that it is easy to tell which page is the current and how many other pages are available. This number line can also be used to change pages with, either by sliding or touch an graphical element illustrating a page.

9: Rearrangement of pictures : *As a child I want to rearrange pictures on a page, so that I can personalise my pages.* From the collaboration meeting at Birken it was experienced that some of the children like to arrange the pictures in their own PECS book. The children make their own system and moves the PECS pictures to the pages they like and in the order they like independently of the pedagogue. This feature should also be available in DigiPECS, and has to be implemented in a way that is intuitive for the child to rearrange the pictures.

This means that some consideration is needed into how to make the rearrangement of pictures intuitive as possible. The required functionality can be achieved in several ways. One way is to rearrange all the pictures after a picture has been dropped to a specific location on the page, much like the way a list is rearranged, when a new element is added to the list. This means that in worst case all the pictures will be affected by the rearrangement. We have chosen a different way of implementing this functionality. To minimise the visual disturbance from the rearrangement of one picture we have chosen to create a swapping mechanism. The child can choose a picture that it wishes to rearrange, and drop this into a position of another picture in the book. Then these two pictures will swap place with visual effect to guide them on what is occurring.

In addition to rearranging picture on the same page, it should be possible to rearrange pictures on two different pages. Once again this can be accomplished in different ways. The child could move the picture out to side of the page to indicate that the pictures should be placed on the next page. However, a small problem occurs when multiple pages are available and the child wishes to move the picture to a page that more than one page away. This means the action of moving the picture to a side will be repeated multiple times. Instead we have chosen a solution that allows the child to drop a picture on top of the image representing a page in the page bar. The child will then be taken to the indicated page where the picture can be dropped into a position of another picture. Here the swap mechanism will be initialised and two pictures will swap across pages.

19: Start secured administration mode : *As a guardian I want a separated and secured administration mode, so that the application settings is inaccessible for a child.* This is main entry point for the user role *guardian* of the system. The motivation for having an administrative interface, is to allow the guardian to alter the settings of the DigiPECS application. As a result the administration interface should be inaccessible for the child, so the child cannot enter this part of the application and by accident change settings. However, since the administration of a PECS book is maintained continuously throughout the day by the guardian, this interface should be easily accessible. How the administration interface is accessed, is the responsibility of the group developing the launcher.

The administrative interface will hold all the information that is significant in administrating the child's DigiPECS book. This entails adding, editing and removing pictures that are shown in the child's part of the application, including adding sound and text to pictures. In addition this is where situations are handled, so it is essential that activating and

altering a given situation is fast and intuitive.

15: Adding picture with camera : *As a guardian I want to take pictures with the camera, so that I can easily add picture of the item in front of me.* From the Birken collaborative meeting we realised that one of the more comprehensive tasks for the guardian is to maintain the PECS book. This includes creating and adding customised pictures to a PECS book that is not already digitised. These pictures usually have to be captured using a digital camera, edited and printed on a computer before they can be physically laminated. In DigiPECS the integrated camera, normally found in a mobile device, can be utilised to capture, edit and share pictures in one step in the administrative interface. This will hopeful reduce in the workload of having to create new pictures for the child's DigiPECS book and allow for a more customised book.

14: Lock picture to position in sentence board : *As a guardian I want to be able to lock a picture to a position in the sentence board, so that I can teach the child about the present of a "I want" picture when constructing sentences.* In the collaboration meeting with Birken a specific feature request was stated. When a child is first taught to construct sentences, a picture such as "I want" or "I need" is placed in the sentence strip of the PECS book, so the child will adapt to using this picture to construct sentences. This is accomplished in order to help the child familiarise themself with a notion of a sentence consisting of distinct parts. In a digitised solution this request can be achieved by locking a given, or a series of picture, to a position in the sentence board. This is also a feature that will be available in DigiPECS by accessing the administrate panel of the application.

This conclude the description of the selected user stories. The complete list of user stories, with a short description, is presented in Appendix A.

8.3 Quality Attributes

A quality attribute is a desired feature or property of a system, that describes non-functional requirements to the application under development. They are used to achieve a set of goals, that may come from different stakeholders with different interest in the product, and can therefore have different meaning to different stakeholders [BCK03]. Quality attributes can help to establish and describe these goals, so the goals become common and understood by all interested parties. Later these attributes can be considered as guidelines when design decisions have to be made, throughout the actual implementation of the system and as evaluation criteria for when the system has to be evaluated.

The development process for this project is influenced by the agile paradigm, as described in Chapter 6. The consequence of this is that no up front requirement specification has been constructed before starting developing the system. One way of ensuring that the project stays on track, is to identify system quality attributes and rate these to reflect the overall goals and quality within this project. Underneath a set of attributes that we wish to evaluate the development of DigiPECS is presented.

8.3.1 Rating of Quality Attributes

The set of attributes is influenced by the definition of system quality attributes found in the literature [BCK03]. However, some of the attributes may be defined slightly different underneath, to suit the goals we wish to aspire to throughout the development of DigiPECS. In addition each attribute is rated on a scale from **not important**, **less important**, **important** to **very important**. The rationale behind each rating along with ideas of how to meet these goals will be elaborated in the following.

Usability *outlines with what ease the user of a system can achieve an intended task, and how well the system provides user support. This includes learnability of the system,*

how efficient the system is in helping a user to accomplish a desired task, and how well the system adapts to the users needs.

This attribute is rated **very important** for DigiPECS. The application is used, by parents which are not necessarily IT-experts. The context may be intense and confusing, as they have to handle children and household at the same time, therefore the system should assist them, such that time spent on reading user manuals is minimised. Hence the UI should be intuitive and efficient and provide feedback when help is needed.

The children may not be able to read and they may be very impatient. We can therefore assume that they will be frustrated, or loose interest, if they cannot find the option they are looking for. Hence the application should be very predictable and intuitive to use. Help texts cannot be used, which requires that other methods are used to assist them.

Maintainability *accounts for with what easy the system can be maintained. This is encompass how easily new requirements will make future maintenance and how well the system can manage a change in the environment.*

Maintainability is rated **very important**, mainly due to the focus of the multi project. The project is posted as open-source and might be used by other people or later projects by AAU students. This requires us to create functionality that can be easily replaced and extended. Consequently, consideration into the overall architecture of the software has to be exercised. Keeping low coupling between components, so single components in the architecture can easily be maintained and updated to meet new requirements, is a key goal of the development of DigiPECS.

A secondary way of meeting this goal is to keep code thoroughly documented. This can be accomplished by making reference documentation via tools such as Doxygen or Javadoc, so classes, public methods and interfaces are commented, as to what functionality these provide, and how these should be utilised. Commenting in-line code may also help future developers, especially in places where the semantics of the code may not be clear.

Modifiability *refers to the ease with which a system or component can be modified for use in applications other than what is was specifically intended for. Can be accomplished with good software practises such as keeping code modular and encapsulated.*

This attribute is closely related to maintainability, as it influences the architecture of the program. In the development of DigiPECS we will strive to keep good software practises, which involves keeping code modular, encapsulated and generic when possible.

Google states some best practises that can be used as guidelines in the development of an Android application. These range from UI design, how to best accomplish high responsiveness and performance to testing practices. These practices will also be taken into account during the development.

As this is a goal that can help us achieve better maintainability, and in addition assist us developers to practice sound software engineering, this attribute is rated **important**.

Performance *states how well the overall system performs according to time and thereby measures the responsiveness of the application.*

As the main user group of DigiPECS is children with autism, it is required that DigiPECS is responsive. The Android runtime environment will force a close down of any applications that are unresponsive in more than three seconds. This behaviour will

ruin the user experience and should be avoided at all times. Therefore some consideration will have to be put into how to handle events, tasks doing heavy duty work, how to build a responsive user interface, thus this goal will be rated **important**.

Security *measures the ability of a program to protect against unauthorised use, while at the same provide services to legitimate users.*

For DigiPECS this has been rated **not important**, as the application does not require that any confidential information is stored. However, there are some aspects that needs to be considered when entering the administrative panel of DigiPECS. This is part of the multi project and the responsibility of another group.

Testability *refers to how well the software is constructed to ensure that testing can be performed on the system.*

The rating of this attribute will influence a number of other attributes in this set. Our application is very user-centric, and we do not know if we have succeeded without testing the system with real users. In addition we are part of multi group working together on a joint project. The independent developed components will need to integrate with other components, hence integration tests across the groups are important for project to succeed. Therefore this attribute is rated **important**.

Portability *describes how well the existing code can be reused when moved to another platform.*

When developing software aimed at Android platform this can be hard goal to accomplish. This is due to the very specific way Android applications are created and how the runtime environment is executed. As the requirement for the multi project states that Android is the only platform targeted in the development, software portability will not be taking into consideration when developing DigiPECS.

However, when it comes to portability on hardware platforms this is another case. Today the Android operating system is encountered on many mobile devices, found in all kinds of shapes, build with different peripheral, and with dissimilar hardware quality. These are obvious inconsistencies, that will have to be addressed to be able to reach the masses. Nevertheless, as we will stay within the scope of this project the optimisation of DigiPECS is focused on the Samsung tablet. Consequently the portability attribute will be rated **less important**.

Each individual rating can be seen in Figure 8.3.1.

Quality Attributes & Ratings				
	Very Important	Important	Less Important	Not Important
Usability	✓			
Maintainability	✓			
Modifiability		✓		
Performance		✓		
Security				✓
Testability		✓		
Portability			✓	

Figure 8.2: Rating of Quality Attributes

These attributes will be utilised throughout the development of DigiPECS. They will be used as guidelines for when design and implementation decisions have to made. No

formal goals utilising quality attributes have been stated for the multi project. It may however, have been of great help in the development of the individual components to have started some overall non-functional requirements to the multi project. These could have been goals to strive for in the individual groups, and could have been guideline from the perspective of the multi project.

Nonetheless, some of the attributes above, are influence by the fact we are working within a multi project developed across several groups. Having to keep the system maintainable and testable are partly derived from multi project perspective, as we know the code we are producing has to be integrated with other component and may be used in future development.

CHAPTER 9

ARCHITECTURAL DESIGN

In the previous chapter we discussed which features to implement, by constructing user stories. We also discussed which quality attributes should be prioritised during the development of DigiPECS. These attributes will dictate how focus is distributed during the implementaion. Recall that both usability and maintainability was rated very important. Important attributes are modifiability, performance, and testability.

Through this chapter we are going to discuss *how* to implement these features. Starting with details about what is required to create an Android project – which file and directory layout needs to be present, what the multi project demands from our application, and the overall architecture for DigiPECS. This will serve as a basis for the implementation and the testing.

9.1 Designing for Android

As we already stated in Section 7.2, developing for Android means differences compared to normal desktop applications. In this section we will have a brief look at different aspects concerning this.

The AndroidManifest.xml is one of the most important files in an Android project. This file defines the version number of the application and minium version of Android supported. It also defines a list of permissions needed, such as we need access to the camera, vibrator, and recording audio. The manifest also holds a list of activities supplied. Recall that an activity is roughly similiar to a window on other platforms.

An Android project also includes a set of folders, containing the application. Thus DigiPECS has the following directory structure:

src/ is the location of all source files containing classes.

gen/ for generated java files. This is for the R.java file, which is a class easing access to all resources.

assets/ holds files concerning the multiproject, which should be included in the application, see Section 9.2.

res/ is all resources for the application. This folder includes several sub folders:

- **anim/** for animations. Not used, as all animations are generated dynamic within the source files of DigiPECS.
- **drawable/** holds small graphic files, icons, and backgrounds (xml files with shape, color and gradient definitions).

drawable-{ldpi,mdpi,hdpi}/ folders holds scaled versions of drawables. For icons the official recommended sizes is (32x32), (48x48), and (72x72).

layout/ holds layout files, describing the layout of components within the application.

menu/ holds menu files, for creation of pop-up menus.

values/ holds files describing content for the application. See below.

The drawable-{ldpi,mdpi,hdpi} folders have been used where feasible to encrease portability accross multiple screen sizes. Some backgrounds have also been designed through shape files, which allows Android to draw the background itself, depending on the size of the object.

9.1.1 Values Files

The values directory in an Android project holds files which conveniently can be used for several purposes. Here we will have a look at how DigiPECS uses these files. All files are xml-files.

First is the colour file. This file describes a set of colours which is applied to the user interface. For example it describes two colours for the background in the sentence board. These are then applied as a gradient. The idea with all colours in one file, is that the colour theme, for DigiPECS, may quickly be changed.

Secondly we also find three strings files here. The first file holds small strings to be placed in various positions through the user interface, such as the application name and button texts. The second file holds the set of longer help messages to be shown. Finally, the third file holds a set of error messages, which is used for informing the user of an error. The basic idea with string files is that the files will be copied and translated into different localised versions, allowing easy translation of the program to other languages. All string files in DigiPECS will be populated with Danish strings, preparing DigiPECS for usability testing.

The last set of files in this directory, is the styles files. These files contain different styles to be applied on the user interface, such as text styles.

9.2 Multi Project Demands

This project is not developed as a stand-alone application for Android – but is a part of a larger project. Therefore, we also have to consider certain demands and considerations into the design and implementation of DigiPECS.

First all activities meant for the children should inherit from `GirafActivity`. This activity makes the screen go fullscreen – that is, process lines above the application will be hidden to the child. It also enables a key combination for accessing the administration interface. Unfortunately this activity also disables menu button on the phone, as an unpleasant side-effect. To re-enable this button, we have our own activity class extending `GirafActivity`. This activity catches all buttons, before `GirafActivity`, and only sends the volume keys to it, as these are used for the key combination.

The multiproject also requires us to have a settings.xml present in the assets folder. This may not contain anything, but is needed before our application is allowed to be installed through GirafPlace. However, in reality these files includes a set of settings and activities we would like to have present in the administration interface.

One icon for the application should be present in the drawable-mdpi folder, which is used by GirafPlace, when displaying the application. When uploading DigiPECS to GirafPlace we also need to specify which user profile settings we require.

Finally the multiproject also offers us a library, which is used to access all settings defined in the settings.xml file. This has been wrapped into a class, which we will discussed below.

9.3 Architectual Layers

In DigiPECS we applied a classical layered architecture, inspired by OOA&D[MMMNS01]. The architecture consists of four layers, as shown in Figure 9.1, together with a collection of exceptions, which are used throughout all layers to communicate messages. The layers are physically created within the source files using java packages, though some of the layers are a collection of multiple packages.

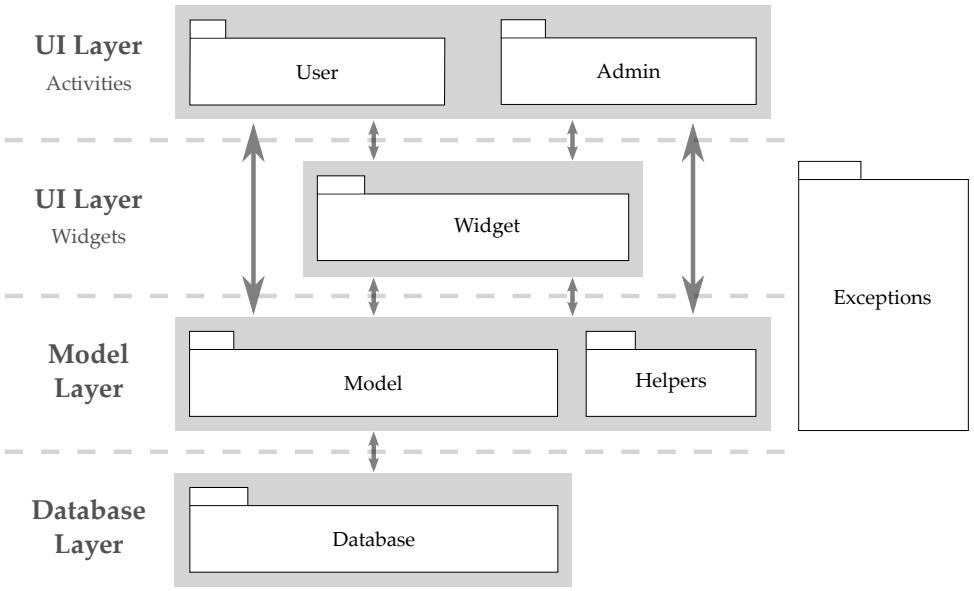


Figure 9.1: This diagram shows the layered architecture of DigiPECS

Activities are the closest point to the user, and thus in the top of the architecture. In DigiPECS we designed two types of activities; activities for the children and activities for the guardians. They are respectively representated in the packages *User* and *Admin*.

Widgets are reusable interface objects, used as building bricks to construct activities. They are small objects like buttons or more complex as a sentence board.

Seperation of the user interface elements assists us is improving the usability of the system, recall from Section 8.3 that is quality attribute is rated very important. It allows us to concentrate purely on designing user interfaces, without concentrating on more functionality related tasks, such as accessing the database through SQL statements.

Both the activities and the widgets need access to third party functionality, like media player or the multi project library for settings.

Models are the only access point, for activies and widgets, to the database. They model all the objects stored in the database. The purpose is to ease creation, modification and deletion of these objects.

Helpers are small helper classes, also with the purpose of easing tasks for activities and widgets. In contrast to the models they ease access to third party functionality, provided by Android or the multi-project.

The model finally needs access to a database. Great effort has been taken to ensure that the Model package is the only package depending upon the Database package, not even the opposite direction applies. However information is sent from the Database to the Model, but only as results on method calls by the Model. This very low coupling between components is a direct consequence of the importance ratings of maintainability and modifiability in Section 8.3.1.

Database consists for classes which hides the raw sql-statements of the database in methods, which is used by the model in accomplishing its tasks.

The seperation in components also have the side-effect that it improves testability of the systems. The best example here is the database layer, where the seperation allows us to accomplish unit tests of it to ensure quality.

The exceptions created within the Exception package; DatabaseException, ImageIsLockedException, DraggableNotViewException, NotOpenException, IDNotFoundException, NotSupportedException, ImageIsEmptyException, NoViewFoundException, ImageIsInvalidException, SoundNotFoundException. Some of these exceptions are explained further within the next chapter.

CHAPTER 10

DESIGN AND IMPLEMENTATION OF LAYERS

In Chapter 9 the architecture for DigiPECS was presented. This architecture is a layered architecture consisting of three layers with low coupling. The low coupling allows us to create components with well defined responsibilities. At the beginning of each sprint our product backlog is prioritised and a number of user stories are selected for implementation. This number depends on two factors. An estimation of a user stories complexity and a estimation of available work hours for the project.

The design and implementation of a user story typically started with a quick design and plan on the blackboard. Afterwards the implementation is started. For complex user stories involving several layers, the implementation in each layer has been performed in parallel. This is possible due to the low coupling and the well defined responsibility of components. On complex sections of code, pair programming have been employed to ensure high quality code and to distribute developing knowledge among the programmers. In the end of a sprint, the user story is tested and unimplemented tasks is moved to the following sprint. This development method do require a fair bit of refactoring which have been done throughout the project period.

In the following the design and implementation of DigiPECS is presented. The layers are presented one at a time. First Section 10.1 presents the design and implementation of the user interface layer. This includes the interaction design for both the DigiPECS book and the administration interface. In Section 10.2 the model layer will be presented. At last the database layer is described in Section 10.3.

10.1 User Interface Layer

The user interface layer in DigiPECS is seperated in two layers: An activity layer and a widget layer. Activities are built using reusable widgets. However, activities are also responsible for configuring these widgets and handling life-cycle related tasks. To accomplish this the activity layer also needs direct access to the model layer. Examples of this relationship are explained later in this chapter.

The activities consist of two different packages; User and Admin. The User package holds the DigiPECS Book interface, for the children, where the Admin package holds the administration interface, for the guardians.

10.1.1 DigiPECS Book Interface

The DigiPECS Book interface consists of only two classes, as shown in Figure 10.1. GirafActivity has unpleasent behaviour relating the the menu button, see Section 9.2, we

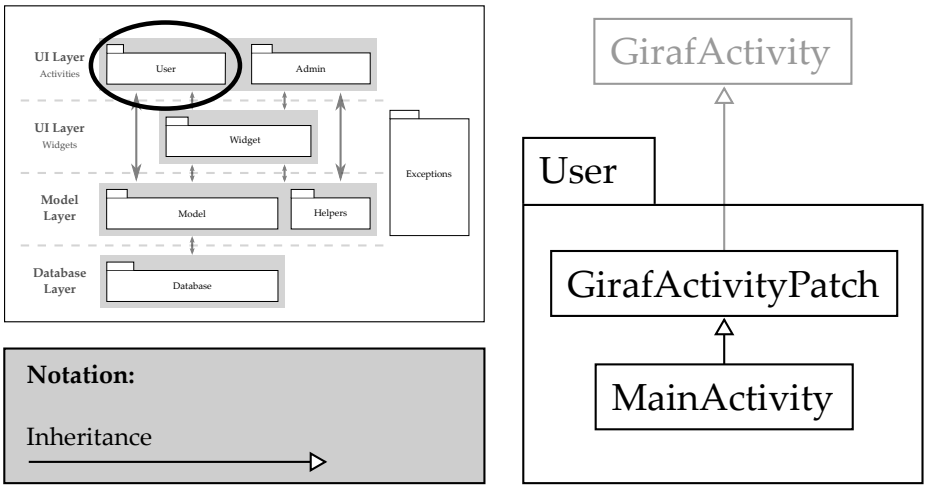


Figure 10.1: This diagram shows the contents of the user package

have corrected this using the GirafActivityPatch class.

Besides this, the only class present is the MainActivity, which is the main entrance point for the whole application. This shows the PECS book with pictures, and the sentence board. An Android activity serves mainly three function: Graphical user interface through a related layout file, setting up classes, and handling life-cycle of the application.

DigiPECS Book Prototype

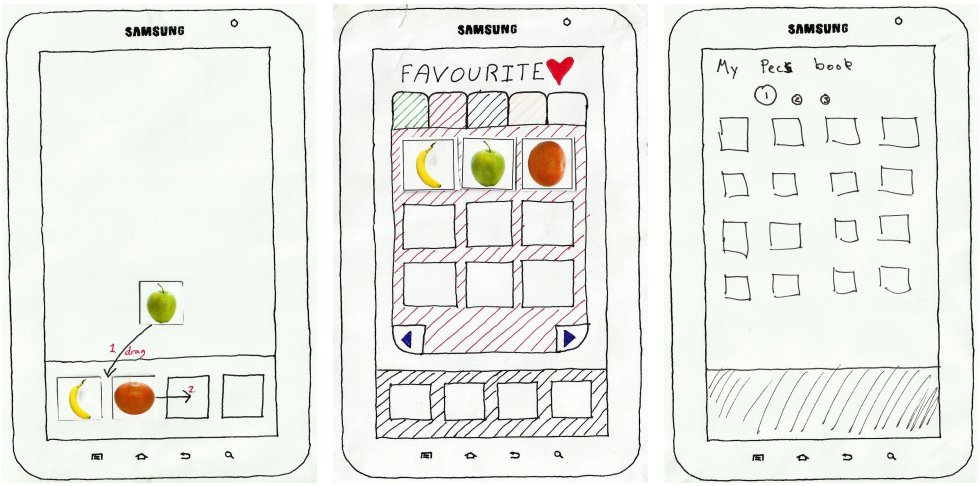
First we shall have a look at the graphical user interface for the children, and how it was developed. The main ideas for this interface were to keep it similar to the PECS book that they are already familiar with, see Section 3.3 – this means they likely would learn how to use DigiPECS quickly. At the same time it has been important to keep the interface simple and intuitive to use, see Section 8.3.

As mentioned earlier we have developed a paper prototype for the child's interface. Paper prototypes are easy and fast to draw, and easy to throw away. Using this method we were able to quickly communicate and evaluate our ideas, until the first single draft screen has become more detailed and futher screens added. Here a few screens will be shown to illustrate some of the design choices made in DigiPECS.

The first prototype screen, see in Figure 10.2a, shows the sentence board, and main purpose, of DigiPECS. The purpose is that the child drags a picture, from a library in top of the screen, into a sentence board field. The picture then sticks to this field. The idea on this prototype is that pictures may be dragged in-between each other, with the result that the other pictures move to make room for the new one.

The final implemented version of the sentence board is similar to this prototype. However pictures may not be dragged in-between eachother, only one top of the other, where they will then replace an possibly existing picture in the field. The implemented sentence board also allows the user to swap two pictures in the board, by dragging them on top of each other. Pictures can be removed by dragging pictures onto a trash can icon, which will be shown whenever a picture is taken from the sentence board.

One of the original, and more complicated, prototype screens, showing the book, is seen in Figure 10.2b. This screen includes nine pictures on a page. More pictures can then either be stored within categories – tab planes in the top, or as pages, using the arrows to switch pages.



a. Drag in Sentence Board b. Drag in Sentence Board c. Drag in Sentence Board

Figure 10.2: Paperprototypes developed for the DigiPECS book

We would have liked to use finger gestures for changing page. The motivation is that swiping a finger left or right to change a page resembles the motion of changing a physical page in a book. Our problem during the development was that the majority of the page is covered with images which are all dragable. Most of the time a gesture to change page would be recognised as a command to move a picture.

Later a new prototype screen was developed, seen in Figure 10.2c. This screen is similar to the actual implementation. First the book contains a 4x4 grid of pictures. Each picture has been made smaller for several reasons. First, the detail level of the pictures seem sufficient with this size. Secondly, it allows more pictures per page, decreasing the possibility that page changes are needed. All pictures in one page allow for a more intuitive user interface, as the child does not have to learn how to switch pages.

The usability is the reason for removing the categories. The idea is that the guardian instead manages situations through the administrator interface, thereby desiding what the child is able to choose from. The page arrows have been replaced with a row of circlular buttons, with numbers. This makes it possible to switch straight from page 1 to 3, without having to go to page 2 first. Only difference from Figure 10.2c to the actual implemention is the lack of text in top of the screen. This has been removed, so that it will not take up space – remember, the children may not be capable of reading.

Implementation

During this implementation description, we will refer to user story numbers where needed. The full descriptions of these user stories may be found in Appendix A.

Figure 10.3 shows the actual implementation of the children interface. Strong colours has been added to make a clear seperation of the different elements within the screen. The colours also aids in making the application more friendly and fun to children.

Pictures are intuitively dragged around on the screen. This emulates the real PECS book, see Section 3.3, where pictures may be moved freely around. When a drag of a picture is initiated a sound is played, according to User Story 11. This sound is a recording of the word the picture symbolises. This aids in teaching the children to pronounce the words themselves. Beside the sound a picture also contains a small text. The text aids both guardians in seeing what picture symbolises and aids the children in learning how to read.

Several drag options are possible in DigiPECS:

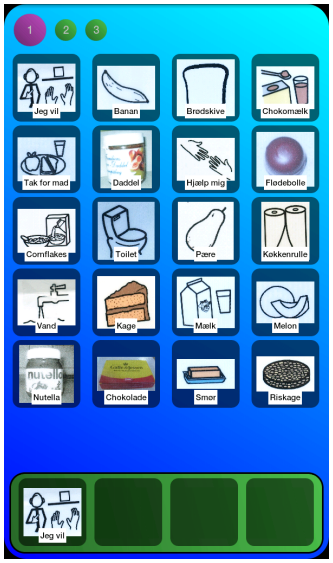


Figure 10.3: The implemented DigiPECS book interface

From book to sentence board (User Story 3): This is probably the most performed drag in DigiPECS. The pictures are taken from the book and droped within the fields in the sentence board. This is used to construct sentences.

Within the sentence board (User Story 1): One picture in the sentence board can be dragged on top of another, in this case the two pictures will swap position within the sentence board. This can be used, by guardians, to teach the child placing verbs in front of nouns.

From sentence board to trash can: A trash can is shown when a picture is taken from the sentence board. If the picture is dropped on this, the picture disappears from the sentence board.

Within the book (User Story 9): Two pictures within the book can be swapped, identically to pictures in the sentence board. Using this feature children may construct their own order and system within the book.

In the very top of the screen is the page buttons. There exist one button for each page in the book. This is useful if there are more pictures than one page can show, according to User Story 8. If dragging pictures touches a page button, the book switches page – this allows swapping pictures between pages.

For guardians, feedback can be given textual and/or visual. However, for some of these children visual feedback is our only option. This accomplish the fact that these children are also visual strong. To provide visual feedback, we have introduced animations.

All dragging options are assisted by small animations. If a picture is not dropped correctly it slides back to its initial position. When two pictures are swapped both pictures slides into their new positions. It is easy to notice what happens on the screen, because nothing comes out of nowhere, everything moves around on the screen.

The menu button on the Android phone is used to open an additional menu, shown in Figure 10.4. This menu contains addional functionality, not essential to using DigiPECS. The two buttons supplied are: “Clear Sentenceboard” and “Play Sentenceboard” (User Story 10).

Figure 10.4: Menu shown when pressing the menu button

The Main Activity

The main activity is the activity controlling all the above functionality – through the use of widgets. On creation of the activity, the class sets up all the widgets with the information they need to work correctly. The widgets used are: AnimationLayout, DragLayout, PageBarLayout, PECSBookPageView, SentenceBoardLayout.

Activity life-cycles was discussed in Section 7.2.2. This is an important responsibility of the activity. Without considering life-cycle the sentence board would be deleted on orientation changes and in other impractical situations. To handle the life-cycle we implemented the following methods:

```
1  @Override
2  protected void onResume {
3      ...
4      mBook.open();
5      AudioPlayer.open();
6      mSentenceBoard.restoreState();
7      ...
8  }
```

onResume opens the PECSBook and AudioPlayer. The PECSBook will open the database connection and fetch the pictures for the pages. AudioPlayer would similary allocate resources. Finally onResume also restores the state in the sentence board, according to the database. If no data is present in the database the sentence board will be empty.

```
1  @Override
2  protected void onPause {
3      mBook.close();
4      AudioPlayer.close();
5      mSentenceBoard.saveState();
6      ...
7  }
```

onPause reverts the actions of onResume. Because both onResume and onPause are guaranteed to be execute by Android, this ensures correct closure of both the PECSBook and AudioPlayer. The Sentence Board is equally guaranteed to have a chance of saving its state.

The MainActivity also handles the menu shown, when pressing the menu button.

10.1.2 Administration Interface

As already introduced, DigiPECS consists of two very different interfaces for interacting with DigiPECS. An interface, introduced above, which allows the children to interact with a DigiPECS book and another for the guardians. As proclaimed in the persona definition Section 8.1 much time is spent creating, updating and reorganising this book on a daily basis. For DigiPECS to succeed, this interface does not only need to be easy to learn and interact with, it also has to be fast and highly effective to use. In this section we describe how we have tried to achieve this. First some general design goals for the administration interface will be introduced. After this, the actual implementation will be presented.

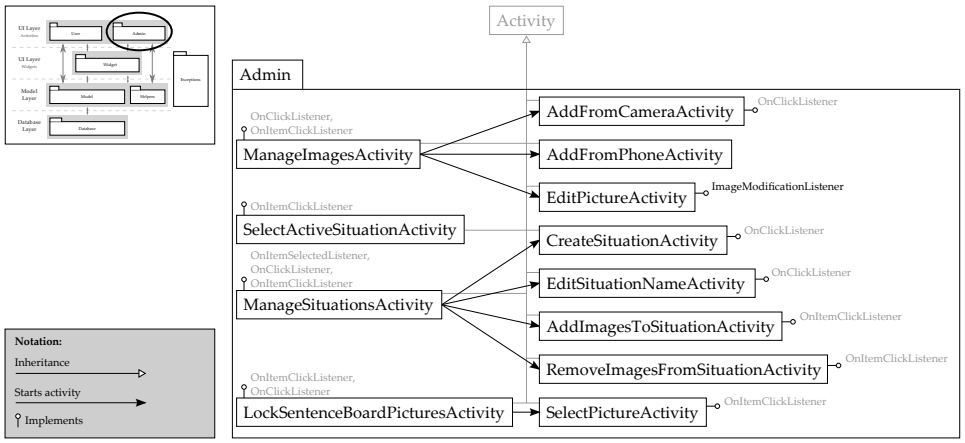


Figure 10.5: This figure shows the architecture of the administration interface

Design Goals

Designing software for mobile devices poses some unique challenge. These devices are frequently not used in a quiet office but in stressing noisy environment which can affect the user's interaction with the system. Understanding this environment, in interaction design called a context, is important to achieve the high usability stated as a quality goal in Section 8.3. Because the persona definitions describe a potential user and its interactions with the environment, these are a useful tool for this task [Ben10]. We learned and observed that a guardian interacts with the PECS book continuously throughout the day. Pictures are added, removed, reorganised or moved to a page showing disabled pictures. These interactions might only take a few seconds but happens frequently. This behaviour pattern not only requires a system that is easy to learn and use, but also a system that can be used fast and effectively in a daycare centre where children and other work tasks can distract the user.

The requirement of high usability and high effectiveness contradicts each other slightly. To achieve high usability, [Ben10] defines the term visibility as important. Visibility is a system's ability to always show the user what is currently happening and what functions the system offers at a given moment. This is a part of the psychological principle that it is easier to recognise things than having to recall them. Following this principle would require us to avoid using the Android options menus, accessible with the menu button. Instead on-screen buttons and text for guidance should be used. On a device with a small screen this results in less, or in some cases almost none, room for the actual content. The lack of room for content will require more scrolling and makes the system less effective to use. A challenge has been to balance these requirements. From all this we can define a few design guidelines for the administration interface:

- Increase the sense of visibility by limiting the use of menus and hidden functionality.
- Much functionality in the administration interface will be concerned with adding, removing or selecting images. There need to be consistency in the way these actions are performed.
- Many of the guardians might have very limited experience with using touch screen devices. Help has to be available at all times, but without sacrificing the efficiency for an experienced user.

In the following the implementation will be presented.

Implementation

The DigiPECS administration system consists of twelve screens, known in Android as activities. Each of these activities handles some amount of functionality requested in Section 8.2. Instead of just describing the screens individually this section will use the individual user stories as a framework. To give an idea of the navigation flow through the interface a navigation diagram is presented in Appendix C

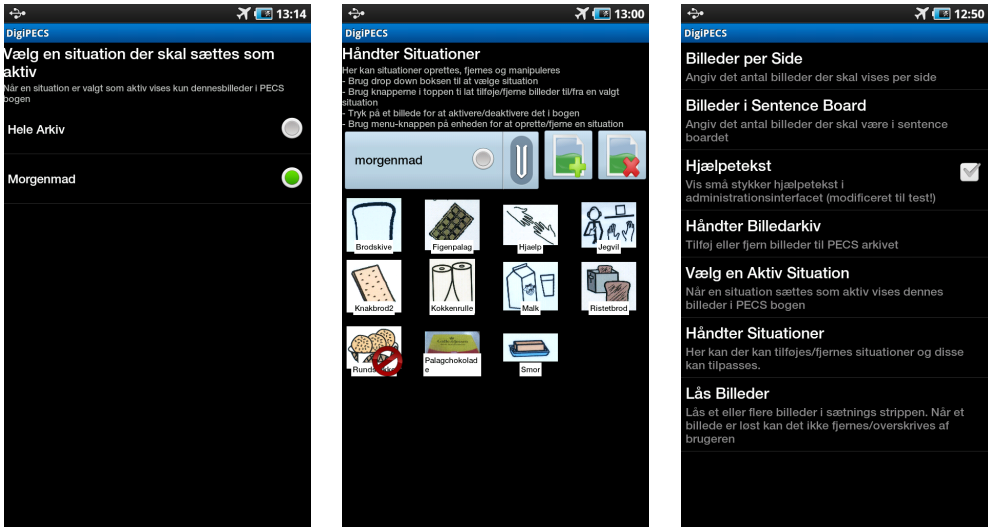


Figure 10.6: This figure shows the interface for `SelectActiveSituationActiity` (left), `ManageSituationsActivity` (center) and the main admin menu (right)

User Story 19: Start secured administration mode In DigiPECS the administration interface is reached by sequencely clicking the volume up/down buttons three times. This access combination is the same throughout GIRAF, and has been defined by the group responsible for the GIRAF launcher.

When entering the DigiPECS administration interface the guardian is presented with the main screen shown in 10.6. The settings on this screen are defined in the `settings.xml` file which is parsed and shown by the group responsible for the shared administration interface. Through this file we can define types like integers, booleans and strings, together with a visible name and description for these. We only have limited control with the ordering and types of settings which can be saved so the most of the administration module have been created from scratch using activities and custom widgets. An interesting setting is the choice to enable or disable help text. If this setting is enabled, a description assisting the user will be shown in the top of all activities. This description takes up a considerable amount of screen space and if used on a small device, or when a user becomes familiar with the system, these can simply be disabled.

User Story 12: Setup Situation DigiPECS supports the creation and handling of situations. In DigiPECS a situation is a subset of the images available in the DigiPECS library that suits a given context. Situations can be easily created, removed or updated with new pictures.

`ManageSituationsActivity` is the main screen for interacting with situations. The content of an individual activity can be shown by selecting it from the drop down box. Buttons for adding and removing pictures from an activity are visible at all times just besides the drop down box. When a situation has been selected images can be activated and deactivated simply by clicking on them. As stated earlier one of our goals is to achieve high

visibility throughout the interface design but on this screen we have had to compromise. Buttons for adding, deleting, and updating situation names have been saved in the options menu, requiring a press of the menu button to show. This is done to increase the amount of screen space available for showing images. This is especially necessary on phones with relatively small screens.

When adding pictures to a situation `AddImagesToSituationActivity` is shown. The activity displays all pictures in the PECS library and allows a guardian to select one or more of these by clicking on them. Clicked items appear with a small plus icon. The design of, and the way the user interacts with it, is reused throughout the interface when the user has to perform multiple selections of images. An example of this is when deleting images from a situation. `RemoveImagesFromSituationActivity` shows almost the same interface. Only the icons, showing if an item has been selected, are changed. In both cases, the changes are saved to the situation when the back button is pressed.

`SelectActiveSituationActivity` allows the guardian to select the currently active situation. The active situation is the one shown in the DigiPECS book. This activity is not accessed from the `ManageSituationsActivity` but directly from the administration main screen. This approach have been chosen for two reasons. First, it allows a guardian to change the active situation with the least amount of button presses, increasing effectiveness. Also, much functionality is already packed into `ManageSituationActivity`. Adding more functionality would crowd the user interface more than needed.

User Story 7: Activate Pictures We have chosen to implement the support for activating and deactivating pictures. The idea behind this requirement is to allow a guardian to easily hide pictures of objects, which for some reason, is not available. In essence this could be implemented in two ways. By allowing the guardian to deactivate a picture in the whole library, including all situations which the picture is a part of, or by allow the guardian to deactivate the pictures in the individual situations. We did choose the later implementation. Often some items, like Nutella, might only be available to a child at breakfast in the weekends. Allowing a guardian to easily activate or deactivate this picture in the individual situation means that the picture would still be active in other situations like "'snacks"' or "'favorites"'. This functionality is actually implemented in `ManageSituationActivity`. When the pictures in a situation are shown they can be activated/deactivated simply by clicking on them. A small icon shows in which state the picture is in.

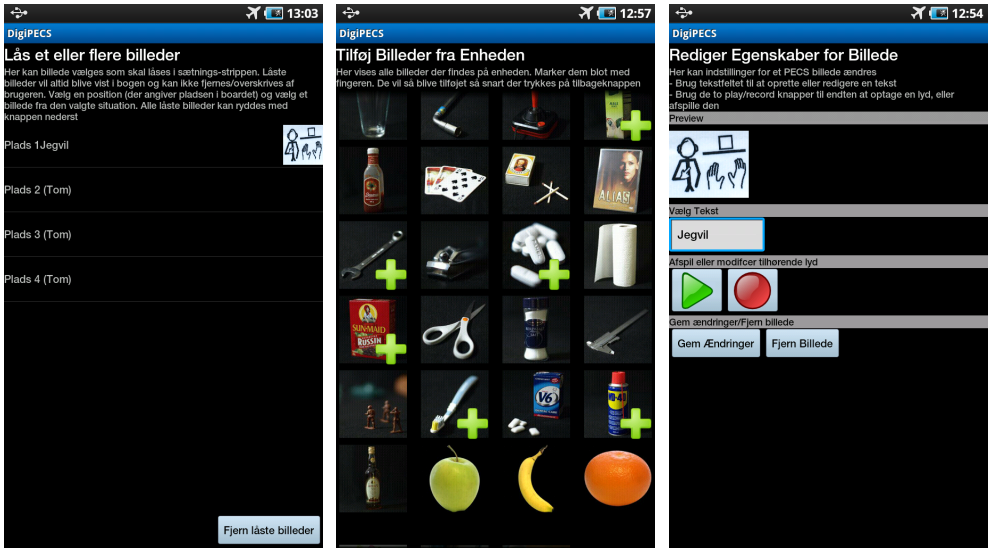


Figure 10.7: This figure shows the interface for `LockSentenceBoardPicturesActivity` (left), `AddFromPhoneActivity` (center) and `EditPictureActivity`

User Story 14: Lock Picture to Position in Sentence Board In DigiPECS pictures, from an active situation, can be locked to a position in the sentence board. This functionality is accessible through the `LockSentenceBoardPicturesActivity`. This activity shows a list containing one item for each position in the sentence board. The item clearly marks if the current position is empty. If a picture is locked, a little preview image is shown. Clicking on one of these items allows the guardians to select a image to lock.

User Story 18 and 5: Modify Pictures/Record a Voice Pronunciation In DigiPECS it is possible to add and modify pictures. `ManageImagesActivity` is the main screen for managing images in the DigiPECS library. This activity shows all pictures, currently in the library, together with buttons for adding new ones. By selecting a picture the user is presented with the `EditPictureActivity`. Here the individual picture can be edited or removed completely from the book. One of the requirements was to make it possible to capture sound to the individual pictures. Utilising the integrated microphone, and the Android API, this is quite easy. A new sound can be recorded or an old one can be played using the buttons in the interface

User Story 15 and 16: Adding Pictures from Device/Camera As described in the persona definitions creating pictures for use in PECS is a comprehensive process. Pictures have to be captured with a camera, located on the Internet or created with proprietary software like Boardmaker. They have to be printed out and laminated to ensure durability. Using a digital device, like a tablet or phone, allows this process to be streamlined considerably. In DigiPECS pictures can be included in two ways. By importing them from the phone's memory card or by capturing them using the camera integrated in the device. This functionality is handled by the following two activities:

`AddFromCameraActivity` allows a guardian to capture an image using the integrated digital camera. To ensure consistency with the rest of the application the user never leaves DigiPECS. Instead the camera preview is shown inside DigiPECS. When the guardian captures the image it is possible to edit text and sound. `AddFromPhoneActivity` allows a guardian to add pictures, located on the device, to the DigiPECS book. This activity lists

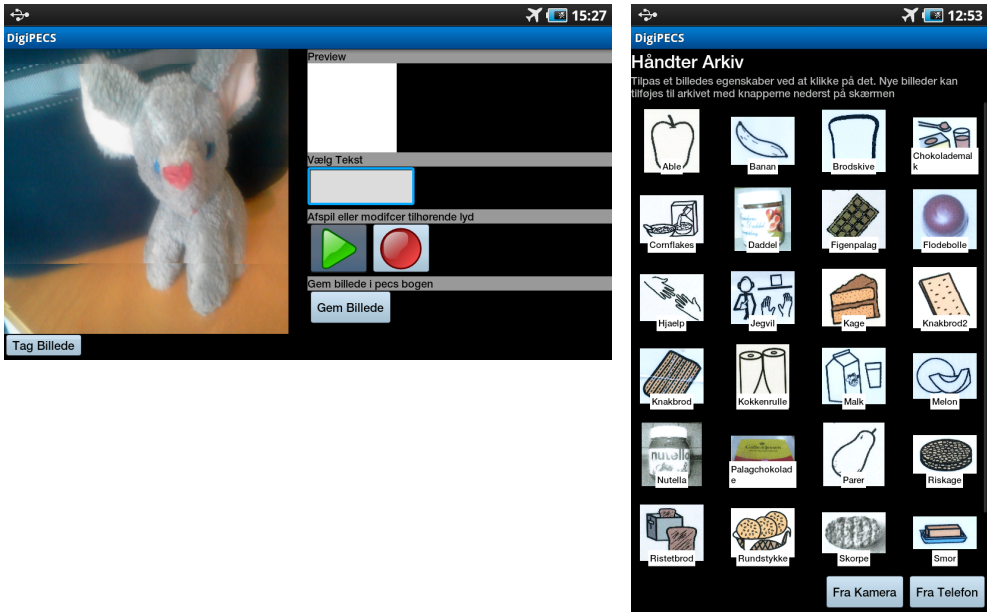


Figure 10.8: This figure shows the interface for `AddFromCameraActivity` (left) and `ManageImagesActivity` (right)

all pictures on the memory card and allows the guardian to mark them for addition by clicking on them.

10.1.3 Widgets

The widget package consists of a set of reusable widgets for the graphical user interface. General widgets which may be used in both children and parent activities that reside within the root of this package, see Figure 10.9. One exception is the drag and drop feature which, due to its number of files, is collected within a separate package. Widgets which are designed specially for either the children or parent interface reside in their respective sub-package; User and Admin.

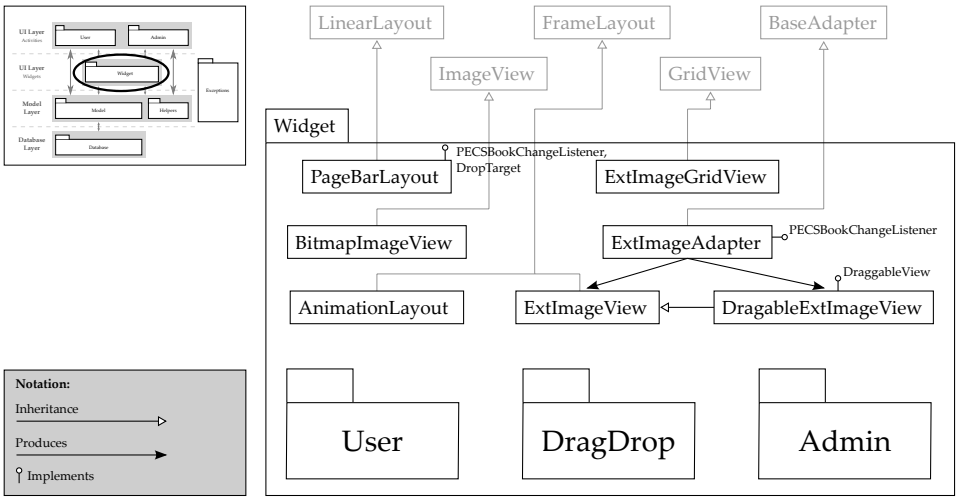


Figure 10.9: The widget packages contains three addional packages, besides its classes

The ExtImageView combines the picture, text and sound into one view, see Figure 10.10. The sound is not visible, but is accessible through a method called play. ExtImageView has an extension called DraggableExtImageView, as the name suggests this view can be dragged. It accomplish this by taking a DragLayout in the constructor and uses this to start the drag, when the user touches the view. The DraggableExtImageView also records from which page it was dragged from, this is used for showing animations.

Figure 10.10: An extended image in DigiPECS

We need to show these ExtImageViews together as sets, for this Android provide views with the purpose of showing data sets. These views extend from AdapterView.

An AdapterView in Android is a view which constructs its children, using an adapter. The two most common examples of these are GridView and ListView. A ListView shows the children below each other in a list, where GridView shows the children in a table-like manner. Both of them implements scrolling by default, so that it may be scrolled if the list extends the area of the screen.

For our ExtImageViews we now need an adapter which can create these for showing in AdapterViews. This adapter is called ExtImageAdapter. This adapter takes a list of PECS pictures (ExtImage) and creates the views from this list. The adapter may produce any sub-type of ExtImageView, the exact type is determined through an ExtImageView factory, which is taken through the constructor. ExtImageAdapter supplies two standard factories, which may be used: NonDraggableExtImageViewFactory and DraggableExtImageViewFactory.

```
1 public static interface ExtImageViewFactory {
2     public ExtImageView createView(Context context);
3 }
```

An Android GridView contains scrolling by default, however we applied our own page structure for the children interface. To disable scrolling in GridView, we have extended this class with ExtImageGridView, which is a non-scrolling variant of it.

AnimationLayout is a small layout which can be placed on top of all other layouts in an activity – it should have all other views as children. Animations may be run in this view. When one or more animation are running, the view catches all touches – that is the user interface is locked until the animation is finished. All animations running in this layer should therefore be very short.

The last two classes within this package are the PageBarLayout and BitmapImageView. PageBarLayout is used for creating the buttons for page switches – these also switch page when something is dragged over on top of them. BitmapImageView is a small view used for animations.

DragDrop Widgets

The drag and drop functionality in DigiPECS consists of one class and four interfaces, see Figure 10.11. Drag and drop functionality is not implemented directly in Android 2.2, and thus has to be programmed within the application itself. To do this we have incorporated

work, by the Android Open Source Project. More specifically from the standard launcher in Android, which do already support drag and drop of icons.

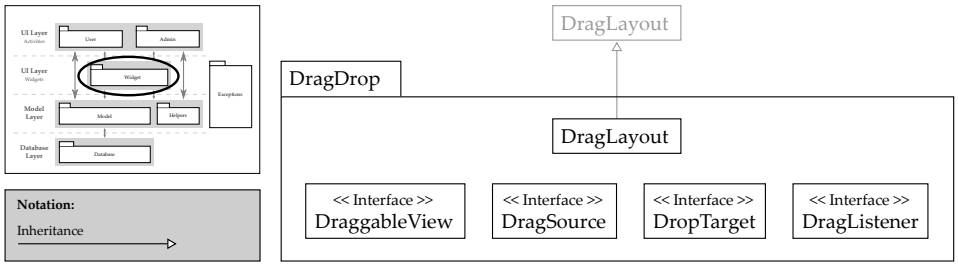


Figure 10.11: Widgets for drag and drop

The interfaces used for the drag and drop functionality are:

DraggableView: This interface dictates three methods; `setDragging`, `isDragging`, and `getReturnAnimation`. The first method will be used by the `DragLayout` to tell the view that it is currently being dragged. Whenever it is being dragged or not, should be determined through `isDragging`. The last method is used to create a return animation for the view, if it is not dropped on top of a valid target. This method may return null – in this case `DragLayout` will provide its own animation to the location the view was originally taken from.

DragSource: This interface can be implemented by any parent to a `DraggableView`, if the parent desires to be notified when the view was successfully dropped. The only method is `onSuccessfulDrop`. One use for this is creating an animation in the DigiPECS Book, when the children successfully drop a picture in the sentence board.

DropTarget: Is implemented by any view that wishes to act as a target for drag and drop – that is, if views should be dropped on them. Three methods are dictated; `drop`, `onDragIn`, `onDragOut`. `drop` is used when a view is actually dropped on the target. It should return true if the drop is accepted by the target and false otherwise. On false an animation is shown, which drags the view back to its initial position. The two other methods are used for informing possible `DropTargets` that a view is being dragged on top of them. This is used to change colour in the sentence board, switch pages in the PageBarLayout, etc.

DragListener: This is implemented by any view that wants to be informed when a drag is started. They should also be added to `DragLayout`, by using its `AddListener` method. Two methods are dictated `onDragStart` and `onDragStop`. These are used for changing colour in the sentence board to inform the user that the view may be dropped here.

The `DragLayout` is a layout which is place in the very top of the activity, around all other views – except for the `AnimationLayout` which will be surrounding this, if present. Lets have a quick look at the typical lifecycle of a view being dragged:

User touches the screen over the view to be dragged. `DragLayout` intercepts all touches and records the position, in case a drag is started later. The touch is passed on to the children, like `DragLayout` was not present at all.

The view calls `startDrag` itself as parameter. This method prepares the view for dragging and sets up the `DragLayout` into drag-mode. It does that by creating a Bitmap of the view (similar to a screenshot), and making the view itself invisible. To the user it will

just seem as the actual view is grabbed. `startDrag` also saves various positioning data needed, and notifies all `DragListeners` that a drag started.

From now on `DragLayout` will intercept and consume all touch events generated. These events can have three major types: down, move, and up. The down was handled by the view itself. The next to happen is move – e.g. the user moves the finger across the screen. Each of these move events results in an update of the coordinates of where the finger was last "seen". It also searches for a possible drop target and calls `onDragIn` and `onDragOut` if necessary.

When the up event rises, `DropLayout` first searches for a `DropTarget` below the coordinates of the touch. If a `DropTarget` is found drop is run, remember this method may refuse to handle the drop. If it accepts, `onSuccessfullyDrop` is called on the parent, if it inherits from `DropSource`. If drop is refused or no view is found, a rewind animation is run, which drags the view back to its original position. Finally all listeners are notified that the drag stopped.

While all the above are happening Android itself calls methods to draw the screen. For getting our Bitmap onto the screen we override the method `dispatchDraw`. In this method we first draw all the children of `DragLayout`, then we manually draw the bitmap onto of the canvas on the coordinates we last seen a touch.

Searching for `DropTargets` happens by recursively searching through all the children for a view that inherits `from DropTarget` and at the same time lays within the coordination of the touch.

`DragLayout` also includes some performance related optimisations. Lets have a quick look at some of them:

```
1  private PointF mLastMotion = new PointF;
2
3  @Override
4  public boolean onInterceptTouchEvent(MotionEvent ev) {
5      ...
6      mLastMotion.set(ev.getX, ev.getY);
7      ...
8  }
9
10 @Override
11 public boolean onTouchEvent(MotionEvent event) {
12     ...
13     mLastMotion.set(x, y);
14     ...
15 }
```

The last motion we have seen is continuously saved in mLastMotion. Instead of creating new object each time we have to save a new set of coordinates, the object is instanstiated only once, and then reused. Similar with other coordinate sets saved within this class.

```
1  private Rect mRect = new Rect;
2
3  private DropTarget findDropTarget(ViewGroup container, int x, int y, int[] dropCoordinates) throws↩
       NoViewFoundException{
4      ...
5      child.getHitRect(mRect);
6      if(mRect.contains(x, y)){
7          ...
8      }
9      ...
10 }
11
12 @Override
13 public boolean onTouchEvent(MotionEvent event) {
14     ...
15     mRect.set(dx - RECT_OFFSET, dy - RECT_OFFSET, dx + mDragBitmap.getWidth + RECT_OFFSET, dy + ↩
           mDragBitmap.getHeight + RECT_OFFSET);
16     invalidate(mRect);
17     ...
18 }
```

Similar this happens to rectangles, which can be representated using the Rect class. An instance of the Rect class is saved as a field, and only created once. Within findDropTarget, this is used to representate the hit-rectangle of a child (that is the coordinate range where the child lays within). It is then easy to check whenever the last seen coordinates lay within this rectangle. It is also used within onTouchEvent to invalidate the rectangle covered by the Bitmap – every time the bitmap is moved, the area needs to be invalidated to inform Android that it should do a redraw of that particular area.

These optimisations are also applied into other classes within DigiPECS.

User Widgets

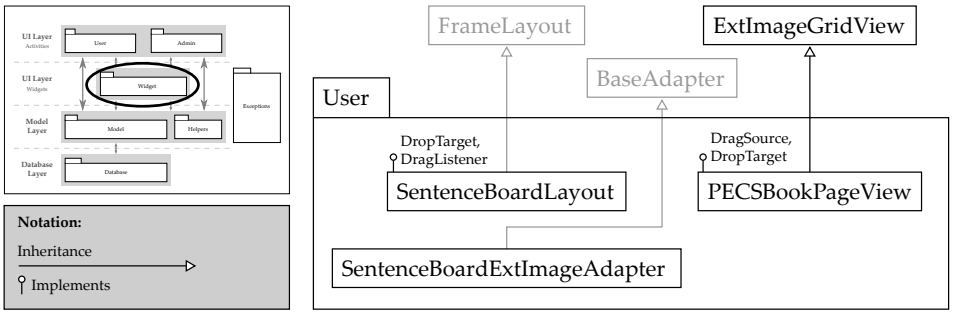


Figure 10.12: Widgets for the DigiPECS book interface

In DigiPECS three user widgets exist, two related to the sentence board and one for the DigiPECS book, see Figure 10.12.

PECSBookPageView: Together with `PageBarLayout` this widget representates the DigiPECS book itself. It shows the pictures, for one page, in a grid. It automatically creates its adapter, when a `PECSBook` (part of the model layer) is assigned to it. It also handles swapping pictures within the book, and the corresponding animations.

SentenceBoardLayout: This widget is the sentence board. It relies on the SentenceBoardList (part of the model layer) as back-end. Among its many responsibilities is to handle all drag and drop involving the sentence board, handling the trash can for deleting pictures, and playing the sound of all pictures in the board. It also passes requests for saving, restoring, and clearing its state on to the SentenceBoardList.

SentenceBoardExtImageAdapter: This is a standard adapter, specially designed for the Sentence Board. It creates two different types of pictures in the sentence board; draggable and non-draggable ones. The non-draggable ones are used when the picture is locked to the board. It also attaches a small vibration feedback to locked pictures, activited when the user initiates a drag on top of them.

Admin Widgets

In the `widgets.admin` package seven custom widgets are used to construct different parts of the user interface. The eight widgets are the following:

ImageBrowser: This is a `GridView` used for showing all images available on the phone's external storage. It includes and manages its own adapter, to be reusable and extremely easy to apply to any layout. `ImageBrowser` should just be inserted into the

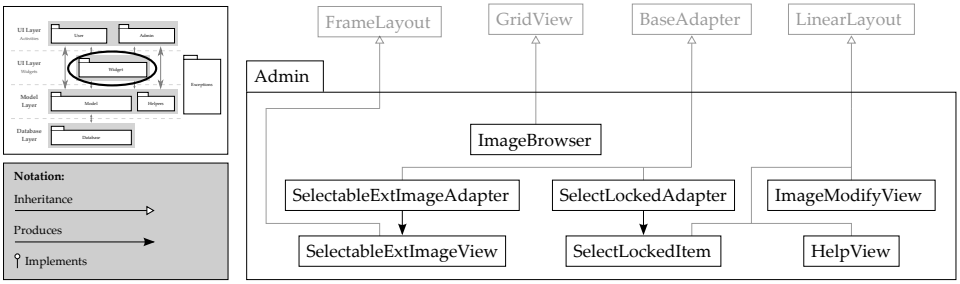


Figure 10.13: Class diagram of the widgets used in the administration interface

layout, and the images loaded by calling the method `loadImages`. The list of selected images may then be retrieved, at any time, by calling `getSelected`.

ImageModifyView: Handles all functionality related to editing a picture in the PECS book; changing text, saving new sounds, deleting the picture, etc. Originally this widget were only designed to allow a user to modify an existing picture, but were modified so it can be used to modify a picture captured with the device's camera.

SelectableExtImageView: This view expands the functionality of `ExtImageView` by putting a small icon on top of it. The icon shows whenever the view is selected or not. Three different icons exist: Add, delete and active. This view is used throughout the administration interface whenever multiple pictures can be selected for activation/deactivation, addition or deletion.

SelectableExtImageAdapter: Adapter for creating `SelectableExtImageViews`. This adapter is used to populate gridviews with images every time multiple pictures can be selected for activation/deactivation, addition or deletion.

SelectLockedAdapter / SelectLockedItem: Adapter and view used to populate a list of pictures locked to the sentence board.

HelpView: Small view for displaying help messages in the administration interfaces. The help view consists of a relatively large headline and some body text. This view is reused throughout the administration activities to show help messages to the user. These help messages can be disabled from the administration main screen.

Some improvements could be made to this package to ensure a cleaner design. When views are used to populate a gridview, or a list, they should be included and managed by the corresponding adapter. This results in cleaner activity code, and if the views should be used in multiple activities, a lot of repetitive code is avoided.

10.2 Model Layer

This layer models different aspects for the above layers. The main purpose of this layer is to ease access to underlying features, both the underlying database and features provided by third parties. The layer consists of two packages: Helpers and Model.

Helpers are small helping classes for accessing Android functionality, saved settings, and handling the first time DigiPECS is launched. The Model package is the classical model of the database.

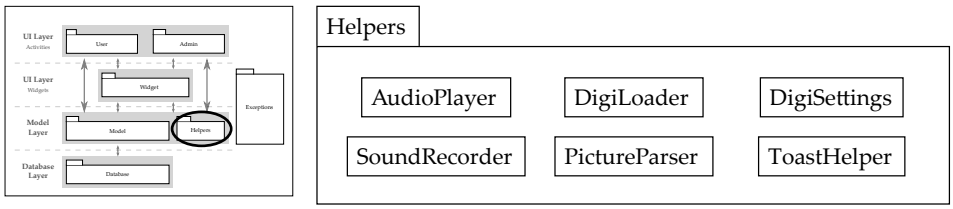


Figure 10.14: These are the helper classes for DigiPECS

10.2.1 Helper Classes

Helper classes are small utility classes programmed to assists the model, or user interface, with functionality that is not directly related to the model. This includes classes for saving and retrieving settings and for capturing sound. This package consists of the following classes:

AudioPlayer This class is responsible for playing back sound files belonging to a PECS picture. The helper class provides a simple play method taking the path to the sound file as argument.

DigiLoader This class has a very specific purpose. The purpose is to allow a user to import a library of pictures into DigiPECS. At the first run of DigiPECS, the method `CheckForImagePackage` searches the memory card for a zip file called pictures.zip. If this file is found, the content is extracted, and searched for a pictures.xml file. This file can specify a list of picture elements, each consisting of a path to a image file, a sound file paths and texts to add to the picture. This file is parsed by the `PictureParser` which adds these pictures to the PECS library.

DigiSettings This class is used throughout DigiPECS to store different settings. The class consists of methods for saving and retrieving the individual settings. Internally, the class uses the `sw6.lib` library for saving and retrieving the individual values.

PictureParser The picture parser is used together with the `DigiLoader` to import pictures, stored in a zip file, into the DigiPECS library.

SoundRecorder This class handles recording of audio. The Android API makes this relatively easy, but using a helper class allows us to hide specific configurations like selection of sound formats and quality from the user interface layer resulting in cleaner code and higher maintainability.

ToastHelper This class is a small utility class allowing the system to print messages to the screen using Android toast messages. The reason for creating a helper class is that it allows us to centrally customise the look and feel of these messages. A special method, `showError`, is used to print out error messages. Besides printing these to the screen, these errors are saved in the Android system log, making it easy to retrieve for developers.

10.2.2 Model

The model package consists of six classes, as shown in Figure 10.15. All is concerned with how to handle and store data. Changes to these classes are often reflected directly down in the database as well.

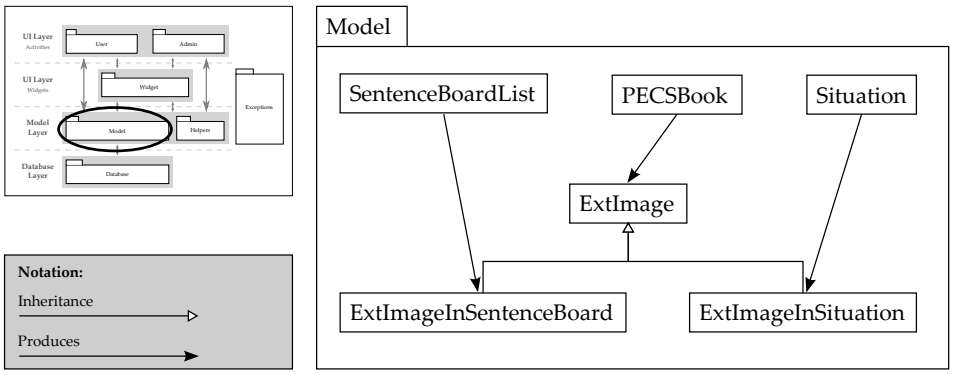


Figure 10.15: This is the Model package in DigiPECS

ExtImage class is a basic class representating a picture in DigiPECS. It contains three basic information about a picture: The image file, a describing text, and a sound file. Not all of these is required to be present.

ExtImageInSentenceBoard extends ExtImage with the functionality that a picture in the sentence board may be locked or not.

ExtImageInSituation extends ExtImage by storing a situation id in the object, and by handling the functionality that a picture is either activated or deactivated within a situation.

All ExtImage classes has a constructor taking an id, which fetches data directly from the database.

SentenceBoardList reflects a list of ExtImageInSentenceBoard which are present within the sentence board. Besides setting, getting, and clearing these, it also offers support for saving and restoring its list within the database. Used for handling life-cycles correctly.

The PECSBook models a book with pages and situations. It is has three different modes:

All: Shows all available pictures in the archive.

AllInSituation: Shows all pictures for a specific situation.

ActiveOnlyInSituation: Shows only the active pictures in a specific situation.

The PECSBook needs to be opened and closed. When opened it contains an open database and cursor, which needs to be closed – this is typically handled through the activity life-cycle.

The two main methods are `getCurrentPage` and `getAllPictures`, which both return a list of ExtImage. This is for getting respectivly one page or all at once. The page and situation are set through `setPage` and `setSituation`.

Another method, to mention, is the `swapPosition` which will swap the positions of two pictures within a situation.

The Situation class models a Situation. From here we can create new situations, update the name, and delete them. Further it have methods for adding and removing pictures.

10.3 Database Layer

The design and implementation of this layer is not directly influenced by implementing user stories, but is designed to provide support for the user stories in the above layers. When implementing some of these stories, a place is needed where data can be kept persistent and which are easily accessible for the running program. One option is to connect to a dedicated server through the data connection on the mobile device, but as it requires

either a SIM card or WiFi configuration, this has not been a viable option and will not be investigated further. In addition *performance* was a quality attribute rated important for DigiPECS (see Section 8.3), so it is essential that the retrieving and storing of data happens quickly.

The logical choice is to save data locally on the device where the application is running. There are several options that offers support for storing data locally on the Android platform. SQLite is a relational database engine, that is fully supported through the Android framework. The framework provides language, package and tool support for managing the database, and easy read and write support for manipulating the database [Inc11b]. Also the Android SDK offers functionality to pull and push files to and from a device, so the database can be managed via a connection to a PC [Inc11c].

Another option for storing data locally on an Android device is to utilise OrmLite. OrmLite is *Lightweight Object Relational Mapping Java Package* that support mapping Java objects to a SQL database including SQLite [Wat11]. The advances of using such a package is that the objects found in your model layer can easily be mapped to the database. However, as a developer you have less influence on the back-end of the database and how SQLite produces the queries and displays the results.

We have opted to use the SQLite database to store data concerning DigiPECS. There a several reasons for this; the model we wish to keep persistent is fairly small (see Section 10.2.2) and should be manageable to map through a relational Database Management System (DBMS), and the language and tool support for SQLite is better supported by the Android framework and well documented. Likewise we wish to be able to configure the back-end of database and customise the results returned after a query. How this is accomplished can be seen in the implementation of the database.

The SQLite Database

The SQLite engine is used in many application today, as it is a small compact DBMS. SQLite is an embedded DBMS implemented as a compact C library, where the full database, including the database schema, is contained within one single file. This means that SQLite is not executed in a separated server process, but is an integrated part of the application that created the database[Hwa11]. In Android each independent application can create a SQLite database. The database is by default private and can only be accessed by the application that created the database [Mei10].

SQLite supports most SQL standards including foreign key constraints. This also includes both language support for defining the database schema and language support for querying and manipulating data in the database. However, the main difference between an ordinary DBMS and SQLite is that SQLite uses a dynamic loosely typing system. This means that the type domain of each relational attribute(column) is not required to conform to a certain type [Hwa11]. In practice this means that even if the domain of an attributes is defined to be of type *text*, an integer can still be inserted to the column in the database table representing this attribute.

Designing the Database

In order to store persistent information concerning the DigiPECS application we need to design a database schema, that represent the model we wish to keep persistent. A consequence of choosing to utilise SQLite, we need to map the model from the model layer into a relational database design. A relational database model uses a set of tables to represent the data to be stored in the database, and the relationship between these data [SKS06]. However, there are some consideration we need to keep in mind, when mapping the model in the database schema. As we are designing a database to be stored on a mobile device with limited storing space available, some consideration will have to be given to the design, to reduce the appearance of redundant data.

In the design of the database for the DigiPECS application, we have chosen to design the database schema using four tables representing the model layer in the architecture. These tables all have table attributes that reflects the data as can be seen in Figure 10.16.

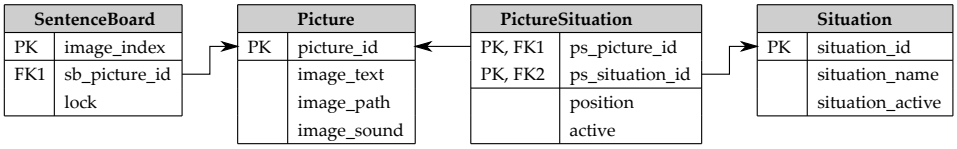


Figure 10.16: Database Schema Diagram

A primary key(PK) will be needed for each individual table, so a row in the table can be uniquely identified. The primary key will in three tables be auto-generated by SQLite, but in the SentenceBoard table the primary key has been defined as the index of the position a picture in the sentence board. This index will always be unique, and thereby we save a table attribute for this table.

Throughout the database design we have used foreign keys(FK) to refer to data stored in other tables. A core example of this usage, is the design of PictureSituation table. The table represents the PECSBook class in the model layer. An PECSBook object will need *the position of all the active pictures for the situation that is currently active* from the database layer. The active attribute in this table, is here to indicate, if a picture is active in the row relationship and not in the given situation. To help model the request of PECSBook class, the table contains foreign keys identifying data, by referring to columns in other tables. With this design it is easy to get information on both a given situation, but also additional data relating to a picture. The foreign key will always refer to the primary key of referenced tables as seen in Figure 10.16.

Another database design constrain to consider, is how to store the image and sound files associated with a picture. In Android neither image files nor sound file should be stored directly in the database [Mei10]. Instead a string should be utilised to store the path to the resource. This string will be parsed as URI that specifies the path to the associated resource.

Implementing the Database Layer

In Section 8.3 it was stated that modifiability was a quality attribute rated *important* for the system. In addition maintainability was rated *very important*. Both goals are reflected in the implementation of the database layer. The database layer is implemented as module, consisting of a database adapter that is responsible for all interaction with the database. This adapter can be seen as the interface to interact with underlying logic. This means, that the underlying logic can be changed as suited, as long as the database adapter still provides the promised and necessary services. This design accommodates the goal of keeping the architecture highly maintainable, as the this layer can be easily replaced by another implementation and be maintained when new requirements are posted.

Some of the work with querying the database and representing the result is however been delegated out to customised cursors. A cursor in Android is an object that is returned as a result of a query to the database. Cursors are pointers to this result and can be customised to suit the intended use - more about this later. In our implementation a factory cursor class is responsible for producing the customised cursors. This design keeps the layer very modifiable, as it is it easy to create your own cursors to suit the data you wish represented. The overall design of the layer can be seen in Figure 10.17

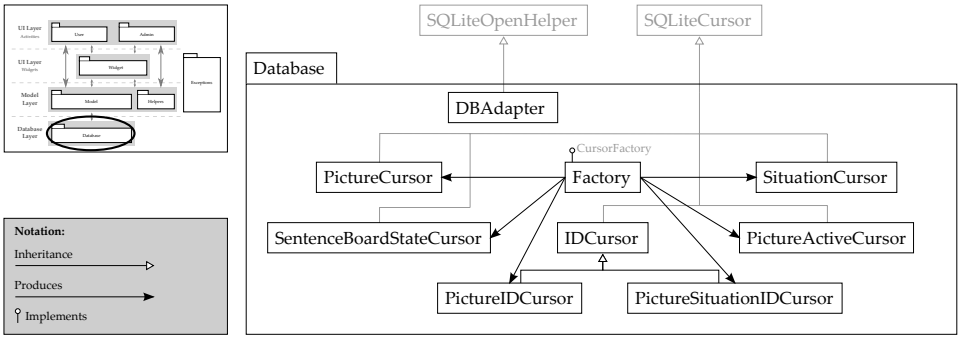


Figure 10.17: Classes and inter-fracture of the database layer

This design has served its purpose in our agile development approach, as the database schema has changed a few times during the development. By keeping the design easily maintainable and modifiable, these changes have happened fairly smoothly, without interfering with the implementation of the other layers.

The Database Adapter As mentioned the database adapter class is the interface to accessing the database layer. The class inherits from the SQLiteOpenHelper class which is an abstract class provided by Android library. The SQLiteOpenHelper class will help get easy access to the database and provides methods that can help creating, upgrading, opening and closing the database.

This database adapter will control all access to the database including updating, adding and removing data from the database and opening and closing the database. Having this adapter will help encapsulate all the interactions with database, by creating methods that corresponds to services enquired by the above layers, making enquires to the database less strenuous. Some of these services are not always easily translated to SQL statement, so this layer also serves an abstraction layer, by providing methods that will encapsulate and hide all SQL queries. An example of this can be seen in the following where the setSituationName method provides support for updating the name of a situation.

```
1     public void setSituationName(int sid, String newName){
2         // Do not change name of Default situation
3         if(sid == 0)throw new NotSupportedException("Default situation(id0) may not change name");
4         ContentValues val = new ContentValues();
5         val.put(DB_Situation_Name, newName);
6         //public int update (String tbl, ContentValues val, String wherClause, String[] wherArgs)
7         mDb.update(DB_Situation, val, DB_Situation_ID + " = " + sid, null);
8     }
```

As mentioned earlier the Android framework also supply package support for easier access to the database. This is also evident in the above example. With the ContentValues class you can assign values to precise columns in the database. In addition the update method works like a normal SQL query, by providing a method, where parameters function as arguments to the query. At it also possible to execute raw SQL queries, but it is recommend to use the helper methods provided by the Android framework, when possible. However, one can argue if the helper methods provided by Android, makes the readability of the code higher.

Creating methods on the database adapter is also a way to control type checking on the values that are inserted, updated and extracted from the database, creating a strong static type checking on the database. In addition some types are not supported by SQLite like the boolean type. Here it is recommended that the values 0 and 1 are utilised for the purpose. An example of where the methods on the database adapter provide an abstraction for such constraints can be seen in the following;

```
1  public void setLockedSentenceBoardItem(int index, int pic_id, boolean lock) {
2          final int locked = ( lock ? TRUE : FALSE);
3          mDb.delete(DB_SentenceBoard, DB_SentenceBoard_Index + " = " + index, null);
4          ContentValues values = new ContentValues();
5          values.put(DB_SentenceBoard_Index, index);
6          values.put(DB_SentenceBoard_Picture_ID, pic_id);
7          values.put(DB_SentenceBoard_Lock, locked);
8          mDb.insertOrThrow(DB_SentenceBoard, null, values);
9      }
```

Here the first statement will convert the boolean argument `lock` to a corresponding integer value. **FALSE** and **TRUE** are two integer constants that respectably are assign the values 0 and 1 in the database adapter to assist the developer of this class to abstract from these two values. The arguments to the method are all used when updating the state of the sentence board.

All queries where the objective is to add, update or delete data from the database are implemented in the database adapter. However all queries that will retrieve data from the database is delegated to customised cursor classes.

Cursors and Factory Class As mentioned the cursors are objects that refers to the result of a query to database. When querying the database the columns to be returned in the cursor are specified. In our design the cursors are returned to the above layers, where they can be used to extract necessary information from the rows and columns in the result. We have chosen to create our own customised cursors to reflect the enquiries from the above layers, so that the cursor object holds the necessary information needed by the reciever. An example of this can be seen in the following, where we introduce the PictureCursor.

```
1  public class PictureCursor extends SQLiteCursor {
2
3      private static final String[] mColumns = { DBAdapter.DB_Picture_ImageText,
4                                                 DBAdapter.DB_Picture_ImagePath,
5                                                 DBAdapter.DB_Picture_ImageSound };
6
7      public static PictureCursor query(SQLiteDatabase mDb, int id) {
8          final String where = DBAdapter.DB_Picture_ID + " = " + id;
9
10         return (PictureCursor) mDb.queryWithFactory(
11             new Factory<PictureCursor>(PictureCursor.class),
12             true, DBAdapter.DB_Picture, mColumns, where, null, null, null, null, null);
13     }
14
15     private int mImagePathIndex;
16     private int mImageTextIndex;
17     private int mSoundPathIndex;
18
19     public PictureCursor(SQLiteDatabase db, SQLiteCursorDriver driver,
20                          String editTable, SQLiteQuery query) {
21         super(db, driver, editTable, query);
22
23         mImagePathIndex = getColumnIndexOrThrow(DBAdapter.DB_Picture_ImagePath);
24         mImageTextIndex = getColumnIndexOrThrow(DBAdapter.DB_Picture_ImageText);
25         mSoundPathIndex = getColumnIndexOrThrow(DBAdapter.DB_Picture_ImageSound);
26     }
27
28     public String getPicturePath() {
29         return getString(mImagePathIndex);
30     }
31
32     public String getPictureText() {
33         return getString(mImageTextIndex);
34     }
35
36     public String getSoundPath() {
37         return getString(mSoundPathIndex);
38     }
39
40 }
```

First all our cursors defines a list of columns to be included within them. This is refected through the static field, in line 3, and the fields in line 15-17. This particular cursor

representates the data used to create an ExtImage in DigiPECS, and thus hold the data; Image text, Image path, and Image sound.

To get an instance of this cursor the static method `query` is used, this method queries the database for an ExtImage with a given id. The query is run in line 10, with the `queryWithFactory` method. This is an Android method, taking a CursorFactory as parameter. For this we create an instance of our generic factory class, in line 11. We will come back to this later. All parameters in line 12 are standard SQL parts like distinct, where, from, etc.

The factory instantiates our cursor using the constructor in line 19-26. The constructor fetches the column indices of all coloumns within it. These indices are to be used through the get-methods in line 28-38.

The whole idea by designing our custom cursors, lays within these get methods. Without them a standard cursor would have a set of methods like `getBoolean`, `getInt` and `getString`. All they take as parameter is an index of the coloumn to return. For a programming using these cursors there is no visual information about which values can be retrieved from the cursor – all this is only defined where the cursor was created.

Having a set of get methods like `getPicturePath` and `getPictureText` provides this visual information to the programmer. All fetch methods in the database adapter returns a particular cursor, in this case say a PictureCursor. At any time during development, a programmer might have this PictureCursor at the disposial. So what information may be extracted from this cursor? – looking at which get methods are available, answers the question. In this case, a `getPictureText` methods are available. This means that the programmer may extract the text belonging to a picture from this cursor.

Recall, that Android requires us to use a factory to create the cursor to be returned when using the `queryWithFactory` method. For this we have created a generic class, named Factory, within the database layer. The idea with this class is, that we provide it with a class during construction, and then it produces that particular cursor types. This happens with `queryWithFactory` calls its `newCursor` method.

It should be mentioned that work on this layer has been difficult to debug. Therefore unit tests have help to verify that the services provided by this layer is consistent with, what it is expected of this layer. The unit tests executed on this layer will be elaborated in Section 11.3.

CHAPTER 11

TEST

High quality code as well as high usability are two features almost every software development team strives to achieve. To attain these features, the code as well as the running system must be tested. Many strategies can be used to test both code and system, and the tests can be performed on the whole system or it can be performed on only parts of the system. How much to test and how many tests to perform depends on the system to be tested. If the system is developed as a Graphical User Interface (GUI) based application targeted at the masses, a usability test is almost inevitable. If the system on the other hand is targeted at software engineers with interaction through a command prompt, the usability is not necessarily all-important. In both situations though, the code underneath is most likely to be wanted to be of high quality, both regarding to not want the system to crash if unexpected events happens, but also to retain the ability to keep the system e.g. maintainable.

DigiPECS is targeted at a specific user-group having special needs with interaction made through a GUI. As stated in Section 8.3, *usability* is an overall goal of the system, rated *very important*, and therefore, the usability of the system must be high. Both for the child and the guardian using the system, it must be efficient to use as it is used on a daily basis. This yields for a usability-test, where both the ease of use, the efficiency, and how the system adapts the needs of the users are tested.

Further, high quality code is desired to conform to the stated wishes of *maintainability* – making it possible for others to develop further on DigiPECS after this semester– and *modifiability* –making code modular and encapsulated. This means that the code must be tested in different aspects.

Throughout the development of DigiPECS varies tests are performed to obtain the above mentioned attributes. A test of the complete DigiPECS is performed as a dynamic black box test consisting of usability testing with two children from Birken and two surrogate guardians. Continuously through the development each user story implemented is tested internally as dynamic black-box testing, conducted by group members to assure the desired functionality is implemented. Critical components of the code is tested as dynamic white-box testing, taking the form of unit testing, to make sure that the running code is executed as expected. As DigiPECS is part of a multi project, an integration test of DigiPECS in GIRAF is performed as well. This is done through a dynamic black-box test, performed by the group members. Each test will in suitable detail be described in the following sections.

11.1 Usability Test

Often usability testing is a cumbersome task, with test preparation, testing the system on a selected group of users, and subsequent processing and analysis of data collected. Particular the process of analysing data collected can be time-consuming in a Video Data Analysis (VDA), with approximately half the time spent on the test is put on data analysis (ref. lektion 11, slide side 3). An alternative to VDA is Instant Data Analysis (IDA), [JKS04]. The main difference is the time spent on analysing data collected during test execution. The test is typically combined with a think-aloud test, where the test person must express all thoughts (s)he has during the test, commenting on the interaction with the system and how it is understood. But instead of recording the think-aloud test, a test monitor and a test logger – both roles taken by a member of the test team – are present during the test, administrating and logging, respectively. Subsequently, an IDA session takes place. The IDA session lasts an hour, and acts as a brain storm and analysis session. Besides the test monitor and the test logger, a third member of the test team – an IDA facilitator – is present during the session. The role of the facilitator is to clarify any uncertainties, and write down and categorise any usability problems encountered. Following the IDA session, the IDA facilitator writes down a prioritised list of problems along with a short description (should take no more than one and a half hour), and lastly the test monitor, test logger, and the IDA facilitator goes through the list to ensure consensus. The entire process of testing and analysing takes only one day – considerably shorter than VDA – and the result is a categorised list of usability problems discovered.

As mentioned, DigiPECS has a specific user-group with special needs. The interaction with the system can be divided into two:

- The interaction with the 'PECS book'
- The interaction with the administration mode

In order to be able to thoroughly test DigiPECS's usability, it is essential to test it on the target group. Especially, as the target group is very different than we are as developers. This way we can encounter problems that otherwise would be undiscovered, as we do not see DigiPECS from the users perspective. Therefore, a usability test is set up to take place at Birken day care centre, where the DigiPECS book will be tested by children with autism. The primary objective with the test is to test if the system is intuitive to use, so the children quickly capture the essence of the book. Besides testing the DigiPECS book on children, the administration mode has to be tested. To gain most knowledge of the efficiency of the administration mode a test should be conducted on a guardian used to communicate with children through PECS. But as the pedagogues in Birken are unavailable to use for the test due to a tight schedule in their work, we have to find other test persons to perform the test. To make sure that the result of the usability test is impartial, two test persons *not* involved in either the development of DigiPECS or GIRAF is used. Further, the test persons are before the test takes place, introduced to how PECS is used, in order to make the test as close to a test performed on a true guardian. The objective of these usability tests is primarily to test if the system is efficient to use, but also to test if the system is intuitive.

11.1.1 Planning of IDA Test – Child

The optimal approach for testing DigiPECS would be to follow the steps of IDA carefully and combine it with a user think-aloud test. However, as the target group are children with autism we have to modify the test to the process. Further, we have a limited amount of hardware at our disposal and we are solely able to be in Birken one day. Consequently, the following modifications along with restrictions of the test is made:

- DigiPECS is intended for children with autism who have no or limited oral communication skills. Thus, a think-aloud test is excluded. Instead, we use two test loggers

during the test to observe problems encountered. We hope that two test loggers find more problems than one. Further, as the child cannot express his/hers emotions orally, and the pedagogue present during the test knows the child and its needs very well, any 'spoken comment' from the pedagogue will be interpreted as coming from the child.

- The children are used to communicate with the pedagogues in Birken. To put the child through as little stress as possible, a pedagogue and a group member will jointly fill the role of the test monitor.
- As only one day is available for us to test DigiPECS in Birken, the test of a real situation – *lunch situation* – can only be conducted on one child. Other tests of usability has to be conducted in a room without a real situation. Still, we hope we can uncover problems even so. E.g. problems of mapping the PECS book to the digitised DigiPECS should be discovered.
- As the children in Birken are not used to choose pictures from different categories, this feature of DigiPECS will not be tested during lunch.

Usability testing is conducted on 2 children with different skills in using PECS.

Test Situation – Child 1 The lunch situation is conducted on a boy of age 6. He is used to use the PECS book in lunch situations, and is the one with most 'PECS skills' in Birken. This boy is confident when expressing his needs and constructs simple sentences using the 'I want' picture as a beginning of most sentences. Child 1 is capable of the words associated with the pictures. To prepare for this test, he is told that the tablet today will be his PECS book. Further, the pictures he has in his own PECS book is imported into DigiPECS. The pictures available during test can be seen in figure 11.1. Whenever the child during lunch wants to request food/drinks he is supposed to use DigiPECS. As his needs during lunch is not known beforehand, a concrete test plan is not made. The purpose of the test is to see if the child is capable of using DigiPECS to get his needs satisfied.

Test Situation – Child 2 This test is performed as an exploratory test, as we can not perform it in a real 'situation'. It is conducted on a boy of age 6, who is familiar with PECS, but has no oral communication skills. He is given the tablet and shown shortly, and during test, how DigiPECS works. The purpose of this test is to see if he can manoeuvre in DigiPECS, and if it is of any interest to him.

11.1.2 Planning of IDA Test – Guardian

In addition to performing an IDA test on 2 children, a usability test must also be conducted on a guardian to test the administration mode of DigiPECS. In this situation we can perform an IDA test without modifications.

Test Situation – Guardians Common for test of both guardians is, that as preparation for the test, the guardian is introduced to different aspects of DigiPECS. Both the administration interface and the DigiPECS book is introduced, and the overall functionality such as camera, locking of picture and editing, addition and deletion of a picture, is presented.

Next, a set of assignments is given to the guardian. The test is performed to discover two things. First, to see if the system is intuitive to use for a person who is not used to manoeuvre in a touch unit. Second, and most important, is DigiPECS efficient to use. Is it fast enough to be considered as an alternative to the analogue PECS book?

Before the test starts, two situations are created in DigiPECS, *breakfast* and *play*, each with more pictures already added.

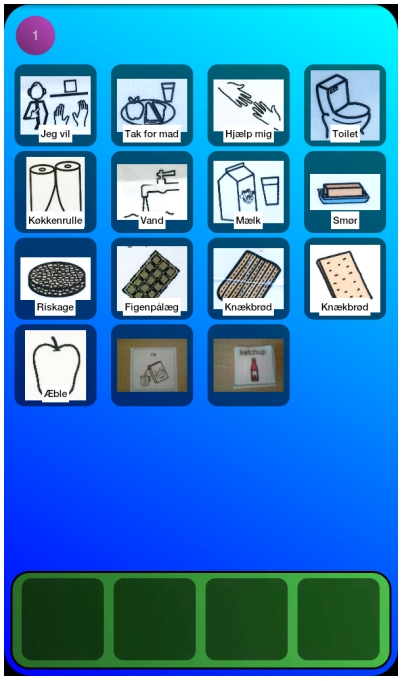


Figure 11.1: The representation of DigiPECS during test of children. Note, rice and ketchup is added during test.

The assignment list given to the guardian contain the following overall tasks:

- Open DigiPECS
- Open administration mode
- Add *milk* to *breakfast* situation
- Add a new picture taken with the camera to the *play* situation
- Add text to the new picture
- Add sound to the new picture
- Create a new situation - *lunch*
- Add *bread* to the *lunch* situation

A copy of the assignment list with descriptions given to the guardian during the test can be seen in Appendix D.

Test Situation – Guardian 1 This test is performed on a male of age 34. He has no prior knowledge to PECS, but has been shortly introduced to how it works. He is familiar with the use of touch phones and Android, and needs no introduction to the basic functionality of the tablet. The preparation for this test is as described above.

Test Situation – Guardian 2 This test is performed on a female of age 25. She has no prior knowledge to PECS, but has been shortly introduced to how it works. Further, she has no experience with the use of touch phones or Android, and is introduced to the basic functionality such as the the buttons on the tablet. To test if the learnability of the administration mode can be considered high, guardian 2 is given the tablet for 10 minutes to make trial and error. By doing so, we can compare the results of guardian 1 and guardian

2, to see if less usability problems is discovered when the test person gets time to familiarise herself with the interface. A condition to keep in mind though, is that guardian 1 has experience in using touch phones, guardian 2 does not have this experience. Further preparation for the test is as described above.

11.1.3 Results of Usability Test

The results of an IDA usability test is usually a list of problems discovered during test. But to substantiate the results, a short summary of each test conducted on the children is provided. A summary of the test conducted on the guardians is not provided, as a test plan for these were given. Instead, some observations made through the test is presented along with selected problems expressed by the guardians. Both guardians managed to complete all tasks, but several problem were encountered through the test. Lastly, at list of all usability problems encountered is presented.

Summary – Child 1 Child 1 was given the tab and told that this would work as his PECS book during lunch. Intuitively, he placed his finger on the 'I want' picture and started dragging it to the sentence board. He immediately captured the essence of DigiPECS and was very curious about it. He tried to touch all the pictures and listened with interest at the pronunciation of each picture. He requested several items using the 'I want' picture followed by a picture of the desired item. Every time he constructed a sentence, he addressed a pedagogue and spoke the sentence - keeping the tablet by his side. When he was given the desired item(s), he dragged the pictures to the trash can. He was shown how to delete a whole sentence at a time, but kept removing one picture at a time as he usually does with his PECS book. He was shown how to get an entire sentence read out loud, but this feature was not of interest for him. Accidentally, he touched the 'back' button on the tablet, and was lead to the 'home' on the tablet. The 'ladybug' icon representing DigiPECS caught his interest, and without help he touched the icon and was back in DigiPECS. At a point during lunch he stated the wish of rice - but the 'rise' picture was not available in DigiPECS. Using the build in camera in the administration mode, the test team quickly made two pictures with pronunciation 'rice' and 'ketchup' and the boy was given the tablet again. Child 1 rapidly constructed the sentence 'I want' 'rice' 'ketchup'. Opening the ketchup caused him problems, but quickly he constructed the sentence 'I want' 'help me' 'ketchup'

Child 1 was very interested in DigiPECS– or the tablet – and acting on an impulsive he exclaimed 'Dejlig' (nice) and hugged the tablet.

The pedagogue - his guardian during test - commented during test, that it would be nice if the tablet could mute. By letting DigiPECS pronounce the words for him he might be 'lazy' and not speak the words him self. But for children with no speech it would be a good idea to let DigiPECS pronounce the words.

Further, the guardian commented the fact that the tablet laid by the child's side during the test. The interaction usually obtained during the use of PECS where the child physically hands the sentence strip to the guardian is lost.

Summary – Child 2 Child 2 was present during the test conducted by child 1, sitting on the opposite side of the table showing great interest in the tablet. At the point where he was allowed to leave the table, he appointed himself to be a test person by approaching the test team and trying to touch the tablet. He started to drag the pictures into the sentence board, and every time he succeeded, he applauded and laughed. He was shown more features of DigiPECS, but he was primarily interested in hearing the words read out loud and dragging the pictures into the sentence board. Child 2 was very fascinated by the sounds the tablet made when he was touching the volume buttons, and he nearly got access to the administration mode. As the test proceeded, he got more and more active, and at a point during test he touched the tablet with all his fingers on both hands very quickly. DigiPECS

locked, with a picture stuck across the sentence board, and the pictures could neither be dragged nor play the audio file. Child 2, though, did not seem to notice it, or he was trying to make it play a sound again – he kept touching it very rapidly.

Observations – Guardian 1 At first, guardian 1 misunderstood the assignment, and tried to add pictures to the sentence board. The test monitor straightened the misunderstanding and the test proceeded. Several times, guardian 1 expressed confusion and doubts. He used the short descriptions for the menus to verify his choices. Many times he requested a 'save' button as he was not sure that changes were automatically saved. He suggested to visualise which situation was active, as he was not sure how to confirm this. The first administration interface he met when pressing the code caused some confusion, as it did not bring him to the administration mode for DigiPECS at once. Trying to deactivate a picture, he was uncertain if he should use the 'fjern billede' (remove picture) button, but decided not to. This would have caused the picture to be deleted from the archive. When guardian 1 should lock a picture to the sentence board, he commented on a misspelling. He requested a way to make a logical representation of the pictures in the archive, as it was not easy to find the pictures he was looking for. When adding a picture with the camera, the tablet was held in the 'wrong' direction, making the picture be represented in the wrong direction. But editing the picture and taking a picture in the right direction caused no problems.

Guardian 1 suggested that when entering a menu in administration mode, a small 'help icon' could appear, and when touched it could show the help messages from the menu button. Further he commented, that when a picture was locked to a position in the sentence board it was a nice feature that it 'growled' when touched.

Observations – Guardian 2 Guardian 2 had the advantage of getting to try the system before the test was conducted. During the trying of the system she commented that the response time from touching to an action took place annoyed her. Further, when trying to edit a picture, she was mistaken the *record* button for a *stop* button. In 'Håndter situationer' she found a misspelling.

During test, guardian 1 expressed confusion of not getting into DigiPECS administration mode when pressing the code. Further, she commented that for a guardian it would be a 'messy' solution to press the code on the volume buttons, and that a child quickly would figure out how to get access to the administration mode. Trying to deactivate a picture, guardian 2 touched the 'fjern billede' (remove picture) button, resulting in deleting the picture from the tablet – without giving any warnings. When adding pictures to a situation, guardian 2 expresses annoyance that the pictures are not represented alphabetically - but comments that it is good that a green 'plus-sign' indicates that the picture is added. Adding a picture using the camera was done in the 'wrong' direction, causing the picture to be represented in the wrong direction in the DigiPECS book. Guardian 2 further commented during adding a picture with the camera that the 'Tag billede' (Take picture) button was placed in an annoying place at the left side - usually it is placed in the right side. Third time guardian 2 accesses the administration mode, she was very annoyed by the code-pressing on the volume buttons.

List of Identified Usability Problems A list of all problems encountered during the IDA usability test is presented in table 11.1. During the test was identified four cosmetic problems, 12 serious problems and seven critical problems. Most problems were encountered during the test of the administration mode, but two critical problems were encountered during the test of the DigiPECS book conducted on a child. These problem should definitely be addressed prior to letting children use the DigiPECS book in every day life.

During the test of the guardians several usability problems were encountered. All problems have to be addressed, but some of the problems also will - if the current user interface

was kept - become smaller when getting used to work with DigiPECS. Due to the wish of making DigiPECS efficient in its everyday use, some of the logical structure in the representation of the administration mode is eliminated. As we have tried to make a system with as little touches on the unit as possible when having to access the administration mode to change settings, many functionalities have to be gathered in one menu, lowering the learnability of the administration mode. It is a trade off we have found necessary in order to make it a real alternative to the PECS book.

The difference in the conditions given to the guardians before test, having guardian 2 getting time to see the system before the test started, does not seem to have an impact on the results. Guardian 2 identified five critical problems while guardian 1 identified only two, and guardian 2 identified seven serious problems while guardian 1 identified six. This could be an indication that even though given a short time to get familiar with the system, many problems in use of it will occur. In Section 14.1 several initiatives are suggested in order to enhance the usability of the system.

Category	**Problem discovered**	**During test of**
Cosmetic	Three misspellings found	Guardian 1 + 2
Cosmetic	'Edit picture' - keyboard annoying	Guardian 2
Cosmetic	'Clear locked pictures' is placed where OK typically is placed	Guardian 2
Cosmetic	Mistaking record button for stop button	Guardian 2
Serious	Lack of mute-button	Child 1
Serious	Less contact (tablet by the side)	Child 1
Serious	Doubt whether 'back' saves changes	Guardian 1
Serious	Lack visualisation of active situation	Guardian 1
Serious	Editing a picture taken with camera meant exit and entering 'Edit picture'	Guardian 1
Serious	Misleading headlines on menus - causing confusion	Guardian 1 + 2
Serious	Unorganised presentation of pictures in archive	Guardian 1 + 2
Serious	Access to administration mode - *not* directly to DigiPECS administration - led to confusion	Guardian 1 + 2
Serious	'Take picture' button places inconvenient	Guardian 2
Serious	Pressing 'Clear locked pictures' clears pictures without warning	Guardian 2
Serious	Toast-messages disappears too quickly	Guardian 2
Serious	User interface responsiveness is slow	Guardian 2
Critical	DigiPECS locked	Child 2
Critical	Nearly accessed Administration	Child 2
Critical	'Remove picture' - deletes picture without warning	Guardian 2
Critical	Code to administration mode - annoyance	Guardian 2
Critical	When changing situation, sentence board does not clear	Guardian 2
Critical	Confusion of direction of tablet when taking picture	Guardian 1 + 2
Critical	'Remove picture' button causes confusion	Guardian 1 + 2

Table 11.1: Test results of usability test

11.2 Acceptance / User Story Tests

As DigiPECS is implemented in an agile approach, functionality is implemented in terms of user stories. When each user story is implemented, a small internal test is conducted to assure the desired functionality is present. These tests are performed mainly as dynamic black-box testing, executed by a member of the group. To extend these black-box tests of

the user stories, some functionality is as well tested by unit testing. These unit tests are presented in Section 11.3.

Two of the user stories presented in Section 8.2, will in this section be pursued with a description of their test design. In Appendix A where all user stories are presented, each user story is as well provided with notes of features to test. It should be noted, that a test design is generated when the user story is taken from the product backlog into a sprint backlog, meaning that only implemented user stories have a test design/test notes. Further, for at user story to be categorised as implemented, it has to pass the dynamic black-box test.

User story 3: As a child I want to move pictures into the sentence board, so that I can create sentences

Approach :

- Dynamic black-box test conducted by a member of the group

Features to be tested :

- Drag and drop
- Sentence board

Pass criteria :

- Pictures in pages should stick to the finger when touched, and stay sticked to the finger as long as the finger touches the screen
- If the finger moves around the screen with a picture 'sticked' to it, the picture should follow the finger's movements
- if the picture is released above a cell in the sentence board, the picture should place itself in the cell (If there is already a picture in the cell, this picture should be replaced with the picture released.)

Fail criteria :

- If a finger touches a picture, makes a movement, and the picture does not follow the finger
- If a picture is released over a cell in the sentence board, and the picture does not place itself in the cell (In both case where the cell is empty and a picture is already placed in the cell in the sentence board)

This user story is implemented during sprint two. It is tested by all members of the group, and is tested several time, both directly after implementation, but also every time changes is made to DigiPECS, to ensure that the functionality is still intact.

User story 14: As a guardian I want to be able to lock a picture to a position in the sentence board, so that I can teach the child about the present of a "I want" picture when constructing sentences. ***Test Design***

Approach :

- Dynamic black-box test conducted by a member of the group

Features to be tested :

- Sentence board
- Locking picture in administration mode

Pass criteria :

- After locking a picture in administration mode:
  - When trying to drag the locked picture away from the sentence board, vibration should happen
  - When trying to drop a picture on the locked picture, nothing will happen

Fail criteria :

- After locking a picture in administration mode:
  - If the locked picture can be moved away/deleted from the sentence board
  - If a picture from the DigiPECS book can replace the locked picture

This user story is as well tested several times. It was implemented during sprint four, and was tested in order to categorise it as implemented. Further, it is tested each time new functionality is added to DigiPECS.

11.3 Unit Tests

Unit testing is usually performed as dynamic white box testing and executed on the smaller parts or *units* of the program. In an Object-Oriented Programming (OOP) language this commonly means testing methods associated with a class. The benefits of carrying out units tests, is that it is possible to locate low level bugs isolated to the specific unit under test and it allows testing of internal part of the program, which may not be caught with other forms of testing. In addition unit tests are normally designed and executed during implementation of the application, implying bugs are located earlier in the development cycle, meaning less cost of correcting the bug [Pat05]. Unit testing is often automated through a testing framework. JUnit is a well-known testing framework for the programming language Java, that integrates well with Eclipse development environment.

11.3.1 Unit Testing on Android

Android provides a unit test framework based on JUnit 3, that allows for testing classes and methods utilising Android specific components [Inc11c]. Test cases in JUnit are implemented as methods that can test units of functionality from the class under test. The test cases are usually named with a prefix `test` and then the name of the method under test. A set of test cases associated with testing functionality on the same class can be organised in test classes. In an Android test project these classes may extend `AndroidTestCase` which lets the test cases get access to elements depended on an Activity Context.

A test case will generally undergo four steps; setup, exercise, verify, and teardown [Gar09]. Setup and teardown is executed before and after each test case. The setup is used to initialise variable and prepare the testing environment, and teardown will ensure the environment is left intact. To assist verifying, a test oracle can help determine whether unit under test has passed or failed. The test oracle is commonly implemented with *assertions*, which JUnit also supports. With an assertion the tester will set up a predicate that is tested when the test case is executed, and if the assertion condition is not meet it will entail that the test case fails [Gar09]. The JUnit framework will log any test that passed or failed and provides a stack failure trace view in case of a failure that offers feedback of where test went wrong.

11.3.2 Unit Testing and DigiPECS

Throughout the development of DigiPECS we have designed unit test cases to be executed primarily on the database adapter class and the associated cursors (see Section 10.3 for

how the database layer is implemented). The class and cursors handle all communication to the SQLite database, where persistent data can saved. The reasoning for designing test cases on this layer is because it has been difficult to locate any bugs related to the database, with only dynamic black box testing used to complete user stories. By executing unit test on lowest layer of the architecture, we ensure that the logical foundations of this layer is working as expected. In addition, with the test cases set up with Android JUnit framework, it possible to bypass the components above this layer in the architecture and test the functionality directly. This has been great help to detect bugs associated with the schematics of the database and the execution of SQL statements as feedback provided by SQLite queries not always are sufficient. The test strategy of these tests has been test to pass.

When designing unit tests for the database layer, we have swayed away from traditional way of creating test classes to match the class under test. As the database adapter class is fairly large, instead four test classes testing functionality on each table of the database have been created. Underneath we will go through an example of such a test class with selected test cases. As most get methods from the database are implemented directly on an associated cursor object (see Section 10.3), the following selected test cases all test functionality related to the situation table in the database and the cursor SituationCursor. First of we will illustrate how the test environment on `DataBaseSituationTableTEST` class is prepared.

```
1       @Override
2       protected void setUp() throws Exception {
3           super.setUp();
4           //Initialise variables
5           mName = "TestName";
6           mContext = getContext();
7           mDb = new DBAdapter(mContext);
8           //Setup test environment
9           mDb.open();
10          mRowID01 = mDb.addSituation(mName + 1);
11          mRowID02 = mDb.addSituation(mName + 2);
12          mRowID03 = mDb.addSituation(mName + 3);
13      }
```

In the setUp method it is possible by inheriting from `AndroidTestCase` to access the context for the application under test, as the constructor of the `DBAdapter` class requires the application context to create a new DBAapter object. In addition all members of `DataBaseSituationTableTEST` are initialised and the database connection is opened. Also three situations are added to the situation table to ensure that there are data in the database to conduct tests on.

```
1   @Override
2       protected void tearDown() throws Exception {
3           super.tearDown();
4           //Teardown test environment
5           mDb.deleteSituation((int)mRowID01);
6           mDb.deleteSituation((int)mRowID02);
7           mDb.deleteSituation((int)mRowID03);
8           mDb.close();
9           //reset variables
10          mContext = null;
11          mName = null;
12      }
```

In the tearDown method the situations added to the database in the setup is removed to leave the application environment in the state we found it. Also all the members of the test class are reset and the database connection is closed. Note that the setup and teardown are reliant on addSituation and deleteSituation are working at expected. This was ensured prior of this implementation of the setup and teardown as can be seen underneath.

```
1       public void testQuery(){
2           //Get a situation cursor with one row
```

```
3           SituationCursor sCursor =  mDb.fetchSituation(0);
4           //Expected number of rows
5           final int expectedNo = 1;
6           assertTrue("01 test fecth situation ",  sCursor instanceof SituationCursor );
7           assertEquals("01 test numbers ", expectedNo, sCursor.getCount());
8
9           //Clean Up
10          sCursor.close();
11      }
```

To be able to query on any getters on the attributes belonging to the situation table a cursor is needed. For this reason this is tested to begin with in the test class. Two queries methods supports this functionality. One for getting all the situations in the table, but there are also cases where a single row from the situation table is needed. To accommodate this functionality a query that returns a cursor with only one row based on an id is needed. The method providing this functionality has also been under test and can be seen here. A default situation is created when the SQLite database is created. The id of this situation is always 0 and the name is found in a string resource file associated with the DigiPECS application which will be illustrated later. This default situation is used to test against in the test case.

```
1       public void testAddSitation() throws DatabaseException{
2           //Get a situation cursor with all rows
3           SituationCursor sCursor = getSituationCursor();
4           //Get the number of rows in this cursor
5           final int countBefore = sCursor.getCount();
6           //Add a new situation
7           final int rowID = (int)mDb.addSituation(mName + 1);
8           //Update the cursor
9           sCursor = getSituationCursor();
10          //Get the count after
11          int countAfter = sCursor.getCount();
12
13          assertTrue("01 test add situation ", countAfter - countBefore == 1);
14          //CLEAN UP
15          mDb.deleteSituation(rowID);
16          sCursor.close();
17      }
```

Now the addSituation method is tested. This is a method that is implemented on the database adapter. In this test case we start by getting the count in the situation table before we add a situation to the table. Then we get a count after we added the situation. The difference between the counts should be 1, which is tested with an assertTrue. After the test assertion has been executed, two clean up statements are carried out to leave the environment intact.

```
1   public void testgetSituationAttributes(){
2           //Default situation created when database is created
3           final int expectedID = 0;
4           final boolean expectedActive = true;
5           //Get the name of the situation from the resources in the application
6           final String expectedName = mContext.getString(R.string.defaultSituationNameDB);
7           //Get a cursor pointing to this situation
8           SituationCursor sitCursor = getSituation(expectedID);
9           //Test getter of this cursor
10          assertEquals("01 test get situation name ", expectedName, sitCursor.getName());
11          assertEquals("02 test get situation id ", expectedID, sitCursor.getId());
12          assertEquals("03 test get situation active ", expectedActive, sitCursor.getActive());
13          //Clean Up
14          sitCursor.close();
15      }
```

In this test case we will test, if it is possible to fetch a situation from the situation table and get the attributes `id`, `name` and `active` of this situation. Again we will use the default situation to test against. Here is an example of how, by inheriting from `AndroidTestCase`, we can get access to resources belonging the DigiPECS application. The name of the default situation is kept in a string resource, where it can easily be modified. We can get access to

this resource by using the context for the application. The assertions will test to see if the values stored in the database are consistent with the expected values.

Test Results

This has concluded the test class setup to test functionality querying the situation table in the DigiPECS database. In total seven test cases have been designed for this class. The complete class can be seen in Appendix E. Similar test classes exist for the three other tables in the database and in total 25 test cases have been implemented for the database layer. The cursors associated with each table are also tested through the test classes. The purpose with the test cases has been to test functionality provided by this layer to other layers, and not internal functionality of the layer.

During the development of DigiPECS unit testing has been of great help to detect bugs and unexpected behaviour associated with this layer. By having automated test cases we have been able to see if any changes to the database have produced any changes to the expected functionality and introduced new bugs. However, during the agile development process, several phases of re-factoring have taken place. This have meant valuable time have been spent on maintaining the test cases throughout the development.

11.4 Multi Project Testing

When individual modules or components are merged and integrated together, an integration test may be performed, to see if the integration between the individual subcomponents have unintendedly introduced errors, to the system [Pre10]. The way the architecture of the multi project is assembled (see Section 2.5), DigiPECS is developed as an independent application providing no direct services to other components. Nonetheless, we have been depending on services provided by other groups, as we are supplying an application that has to be accessible and fully functional in the GIRAF environment, hence integration of components and modules has been of great significance to the development of DigiPECS.

Throughout the development of the multi project, individual groups have released functionality after no predefined or jointly specified time-line, where new releases have been published when a stable release has been ready. Therefore informal integration tests have been conducted throughout the development of DigiPECS, to ensure that our application integrates with subcomponent we have been reliant on. This integration has occurred in the beginning of a new sprint in our development cycle, where adjustments to our project have been conducted if needed. The reasoning for integrating with other components only once every fortnight, has been to provide a stable environment in the time period of a sprint. If any issues has been found in the integration, these have been reported and addressed in one of three ways. By discussing them on leader meetings, by taking the issue directly to the group members or by reporting it on our issue tracker.

11.4.1 Joint Integration and System Testing

However, since DigiPECS is developed as part of a larger project, it is also essential to test if the interaction and integration between all the subcomponents of the multi project are conforming to the joint requirements. Towards the end of the project period a unified integration and system test performed across all groups have been necessary, as to test if all the developed components also conforms to the requirements of the multi project. A test group, assembled by one group member of each group, has been appointed responsible for the multi project test. This is the first time all the components of the multi project are tested in a combined test. The test is conducted over two sessions to save valued time nearing the project closure. In the first session a test plan will be defined and send out to all members of multi project team and in the second session the test plan will be executed.

Beforehand an integration between the administration and launcher group and their depended subcomponents will take place, where any problems and bugs in the merger will be identified. The following will describe the purpose with the multi project test, how it was assembled and the concluding the final results.

Purpose of The Multi Project Integration & System Test

The integration & system test is conducted to test if the complete GIRAF system behaves according to the system definition for the multi project. The purpose of the test is to determine to what extent the different modules successfully integrates. The main approach to this, will be testing the complete life span of GIRAF and how it interacts with the dedicated developed applications. This includes installation of GIRAF and the retrieval and installation of client applications through GirafPlace.

The test is carried out by assembling a test plan utilising a dynamic black box test strategy. To ensure that the complete life span of GIRAF is covered in the test, the design of the test plan is based on the architecture of the multi project see Section 2.5. By following the architecture we ensure that all subcomponents that have to be integrated to have a fully working system, is covered by the test. The test plan will specify a number of test cases that follows the same pattern describing the following; *Identifier*, *Program/Module/Object Under Test*, *Test Objective*, *Test Conditions and Intercase Dependencies*, *Test Procedure*, and finally *Test Results and Observations*.

What is not covered in the test plan are isolated tests of the individual subcomponents of GIRAF. The requirements for executing the integration & system test are that the components that have to be integrated in the test, are in a stable release. This necessitates that tests to accommodate this, have been performed beforehand and have all passed. Testing the actual functionality of individual modules is done by groups individually. The test will be conducted by testers who are experienced Android users and developers, that know where UI elements are prone to fail.

The Test Plan

As mentioned the test plan of the multi project integration & system test has the main purpose of testing the complete life span of the GIRAF based on the architecture of the multi project. With this in mind the test plan has been created to accommodate this intention. The test cases have been designed as individual steps in this process, from the first time GIRAF is downloaded and installed on a device, to uploading new applications on GirafPlace and installing these and then ending by deleting these applications again. In between these steps we will test if the administration settings works as expected, that applications are filtered correctly according to the user profile and the life-cycle of activities in the applications behaves as expected by handling Android specific buttons. Each test case in the test plan will be presented here, but to keep the description of the test plan to a minimum, the test cases presented here will only describe the *Test Objective*. To see the full test plan, please refer to Appendix F.

TC00: Ensuring GIRAF applications can be deployed to GirafPlace, and that non-GIRAF applications and corrupted GIRAF applications are rejected.

TC01: Testing initial installation of GIRAF, and that the user is presented with necessary welcome dialogues.

TC02: Test that applications can be installed on an Android devices using GirafPlace. This includes that the launcher and applications are able to use and store settings in the Lib library, as well as abiding to the navigation flow for opening the administration mode.

TC03: To test upgrading applications to GIRAF with new settings for the admin panel and ensure these are shown on the device, as updated applications that can be installed.

TC04: To test the back button behaviour of GIRAF client and the applications.

TC05: To test the home key behaviour of GIRAF client and the applications.

TC06: To test whether applications can be deleted from GIRAF via GirafPlace.

TC07: To check if applications are filtered on the client through GirafPlace according to user profile settings.

This concludes the test plan and the accompanying test cases.

The Test Devices

To be able to execute the test plan a set of devices are needed. The optimal solution would be to execute the test plan on every device that utilise the Android OS. However, as this is an infeasible option, a subset of Android devices that is representative for the intended user will be chosen. A set of devices were handed to multi project group at the beginning of development. Four of there are chosen to be test devices and are as follows; Samsung Galaxy Tab, HTC Hero, HTC Desire and HTC Wildfire. This subset may not fully represent the phones of the proposed user, but we feel the set is justified by being diverse. The four devices span from the Samsumg Tab's 7 inch screen down to the Wildfire's very compact 3.2 inch screen. The devices are also having different hardware buttons, where some of the required Android buttons are implemented as software buttons as the case with the Samsung Table and others have hardware buttons. The CPU's of the four devices also span over a significant range. From the Samsung Tablet and HTC Desire's 1.0GH CPU to the HTC Hero and Wildfire's smaller 528 MHz CPU. All the test devices are installed with Android 2.2 - a requirement of the multi project.

The Test Results

The overall test result was partly successful, as it showed the subcomponents all integrated according to plan. In addition the execution of GIRAF, GirafPlace and the dedicated applications also conformed to the system definition of the multi project. The test showed it was possible to download GIRAF on to a device, and install and execute uploaded dedicated applications from GirafPlace in the protected environment of GIRAF. It was also possible to enter the administrative interface and change settings in an application handled by the Lib library. Newly updated applications were accessible through GirafPlace, and the applications could successfully be filtered on the device, so that applications specifying certain conditions according to the user profile settings were not shown in GirafPlace on the device. All applications were successfully deleted again.

However, several issues were identified in executing the test plan, concerning all groups, including DigiPECS. Nonetheless, it was still possible to execute the test plan and get through a complete life span of GIRAF. Most if these issues were related to GirafPlace and incorrect behaviour of using the Android buttons. Some issues were also discovered which were not enclosed in the test cases. Most of the issues were concerned with incorrect handling of the Android activity life cycle or incorrect usage of icons interfering with the user experience. An example of this was that on the HTC Wildfire an icon in GirafPlace was so large it could not be read on this device, hence the button's intended use was disguised from the user. The issues were all reported using the issue tracker on the hosting site of the multi project, along with steps to reproduce the issues. This means that all members of the multi project have had a chance to see what was reported from the test and take the necessary actions to correct the issues. The following will list the issues found in the test.

Issue 43: Application state is saved when entering admin mode, then exiting to launcher. Opening the app again opens the administration panel. (TC05)

Issue 44: Invalid settings.xml not detected on GirafPlace. (TC00)

Issue 45: Rotating the phone when editing admin setting, clears content. (TC01)

Issue 46: Cursor not placed to the right in edit text fields in admin.Detected only on HTC devices. (TC01)

Issue 48: Refresh context menu in GirafPlace has too large icons. (TC02)

Issue 49: GirafPlace list is not updated properly on the device, when newly updated applications are available for download. (TC03)

Issue 50: Icon guide lines not followed. (TC03)

Issue 51: An updated application is shown in the launcher twice. (TC03)

Most of these issues were addressed before the project hand-in and unresolved issues can be followed with the issue tracker. Issues relating to DigiPECS were concerned with not following icon guide lines. This has been resolved, by ensuring that icons are accessible in high, medium and low resolution format in the correct resources folders. It also has to be noted that DigiPECS was still under development, when the test was conducted and therefore had to be excluded from a few of the steps in the test. Some unreported issues were also discovered with DigiPECS relating to integration with Lib library, but these were resolved by updating to the newest release of Lib. The test was performed again after DigiPECS was closed for further development, both to have a complete rerun of the test but also to test if issues identified in the first run concerning DigiPECS had been address correctly.

Part IV

Closure

CHAPTER 12

PROJECT CONCLUSION

In this chapter we conclude on the result of this project. This conclusion will consist of two parts. First part is a dedicated conclusion of the DigiPECS project. Secondly a conclusion will be presented for the multi project.

12.1 DigiPECS

Throughout this project, we have strived to digitise the PECS book in a way that would be as easy and efficient to use, as PECS is today.

This has been accomplished, as a part of a multi project, where four groups have worked together to develop a system aimed at children with limited mental capabilities due to handicap or age. As part of this multi project we have been developing an assistive tool, for children with autism – a subset of the overall target group. Many children with autism have problems communicating verbally, and utilise PECS to communicate with. To develop this application, the following problem statement was defined:

"How can we design and develop a digitalised solution of the PECS book, which is to run within GIRAF, used to assist children with autism, in communicating with their communication partner?"

In order to answer this question, we had to obtain a thorough understanding of the target group and how PECS is used. This is accomplished partly through reading, but mainly through a collaboration partner *Birken*. This has resulted in a preliminary system definition stating overall requirements.

Having obtained the knowledge of *who* to develop *what* for, we then defined a set of user stories describing the features to be designed and implemented in DigiPECS. These user stories were used in the planning of the development to help construct task, for implementation, and to keep focus on the end users. This focus was accomplished by prioritising the user stories to ensure that the most important features were implemented first.

In addition non-functional requirements of the system were defined, utilising quality attributes. These were rated, according to their importance, to obtain overall goals with system. Two attributes were rated *very important*; *Usability* – due to the target group, and *Maintainability* – due to the possibility of others continuing the development of DigiPECS.

In our effort to achieve a system with high *usability*, different initiatives were taken. For the DigiPECS book aimed at the children, we have made prototypes to discuss with our collaboration partner. Further, we have tried to map the physical PECS book to the tablet, making it as intuitive for the child as possible. To test if this was accomplished, a usability test was conducted on two children in our target group. The results showed that DigiPECS indeed *was* easy and intuitive for the child to use, though problems were

encountered.

For the DigiPECS administration, aiming at the guardians, prototypes were also drafted during internal discussions within the group. Trying to design an interface proving very efficient for the guardians, resulted in the usability test revealing trade-offs in the different aspects concerning usability. Trying to implement a very efficient use of the administration, meant that navigation behaviour became more cumbersome to accomplish, than intended, and that learnability, to some extend, was reduced. This was confirmed by the usability test, conducted with two surrogate guardians. A part of the usability was lost, and several problems were encountered in the administration mode.

Several initiatives were taken to keep the system *maintainable*. First a layered architecture was chosen, to minimise the coupling between components within DigiPECS. Low coupling mean that components of the system may be redesigned or rewritten without the need to apply changes to the full application. All the code has also been documented, through Doxygen, to ease the job of the maintainers. In addition rating *modifability* important had a positive influence on maintainability. To fulfill this goal functionality has been encapsulated in modules, and classes kept generic and reusable when possible.

Two additional attributes were also rated *important*; *performance* - keeping the system responsive and preventing a close down, and *testability* - to ensure the system can be tested to verify that the goals have been achieved.

Performance is attempted to be achieved using several means; reusing objects in intensive operations, optimising database queries and only extracting the required data from the database. *Testability* is accomplished through keeping components clearly separated, with the layered architecture, making it possible to locate and isolate issues. It also made unit testing of the database possible.

Testing was used to evaluate the quality of DigiPECS and locate issues to correct. Different test strategies have been employed throughout the development; usability testing the usage of the graphical interface, acceptance testing of the functionality, and unit testing of the underlying database.

Due to the development method, inspired by *Scrum* and the agile paradigm, we have been able to constantly adjust our system to changed and additional requirements. Further, it has allowed us to solely make an analysis of functionality that is implemented. We did not waste time on up front analysis and design of features, that most likely would have changed throughout the project, due to the multi project aspect or obtainment of new knowledge. In addition, the method has help planning and maintaining an overview of the development process.

The development has been an interesting and exciting process, seen from a software engineering perspective. We, as developers, have enjoyed the context and setting posted in project proposal. During this process we have accomplished to construct DigiPECS as a part of a multi project. This have encompassed phases of software engineering embracing analysis, design, implementation, evaluation and integration with other components. The result is DigiPECS, which has proven to be a usable digitised assistive tool for children with autism, that can be utilised in the context of GIRAF.

12.2 Multi Project

The aim of GIRAF was to make a safe environment for children with limited mental capabilities due to handicap or age, and a way for their guardians to administrate how and what the child should be offered of functionality. Initial requirements stated that global settings should let a guardian filter what to install, as well as what should be shown to the child. The guardian should control these settings in a secured mode, inaccessible for the child. Effort should be put on making GIRAF, GirafPlace, the administration interface and applications both *easy to use*, due to the target group, but also to make sure that the software should be kept *maintainable*, in order to allow further developments by others at a

later stage.

To assure that GIRAF is applicable for the target group, a test of the system after integration was performed. This test was partly a success, as issues were found, helping the sub-components in improving integration. All identified issues were reported with the issue tracker at our Google Hosting Project page. Some of these issues are already resolved, while others were merely identified. Identification, though, provides a mean to improve the GIRAF in later developments.

The tests showed that the groundwork for GIRAF is implemented, as it is possible to fulfill most the requirements stated for the multi project. The combined project and its resources are well documented and available and posted for others to continue the work. Hopefully, others will embrace this challenge and continue where we left off.

The outcome of the multi project has been to gain experience with the challenges of developing a larger organisational system. Challenges concerning development strategy, communication, documentation, unity as well as integration are all faced during this project. In next chapter, these challenges are approached, and suggestions of initiatives to take for improvements, in a later developments, is discussed.

CHAPTER 13

MULTI PROJECT COOPERATION

In this chapter, a discussion concerning the multi project and the appertaining challenges met during development will be presented. Being several groups, developing a single system, requires communication and coordination between the individual groups. In developing GIRAF, this aspect has been deficient. In the following, we will try to reflect on why this is the case and what improvements could be made, to avoid this in future projects.

As mentioned in Section 2.6 meetings were held in the leader groups every second week. The goals with these meetings were to define the requirements to GIRAF. However, it has been difficult to agree on the target users and the overall objectives for GIRAF. We believe this is due to several factors. Prior semesters have been focusing on working in individual groups. We are not used to interact between these groups, but rather to have a competitive mind. Every group has its own way of tackling a project. Some groups prefers analysing and understanding potential users before an implementation is made, while other groups are focusing on developing large amounts of high quality code.

Developing a project in cooperation requires at least some groups to compromise on their practices. In this project, these compromises were never made. The result has been a continues discussions to define a user group and a set of requirements for the system which satisfied everyone. Instead the requirements have been ambiguous. A small example; the FACTOR for GIRAF defines the target user group as: "Children with limited mental capabilities...". This statement is not consistent with the global settings *Can hear* and *visual impairment*, which are physical handicaps, not mental handicaps.

This example points to a more general lack of shared analysis for the whole project. Some of these issues could probably have been avoided if a joint development method was chosen. This method should be used to help structure the project. A development method typically defines a system development life cycle embracing deadlines for deliverables such as system definition, stated requirements and implementation of individual features. The use of a proper a development method would further allow us to better judge and prioritise the requirements for the system. Additionally to have a common development strategy, making an overall visual overview (in the style of a Scrum board), to assure that everybody knows the status of the project and which features are (to be) implemented. This overview should be available for all members of the multi project either as an online document or a physical board placed accessible for all.

Further, to enhance the cooperation between the groups, a supervisor or a semester coordinator could be appointed to serve as chairman of the multi project. This chairman could be responsible for setting e.g. superior deadlines, and these deadlines as well as meetings could be scheduled in order to assure that all developers were able to attend.

CHAPTER 14
FUTURE WORKS

Throughout the development of DigiPECS- our contribution to GIRAF- some potential opportunities for enhancements have been identified. In this chapter, further improvements for DigiPECS are presented, as well as few improvement suggestions for GIRAF.

14.1 Future Works for DigiPECS

Though DigiPECS is tested on the identified target group and has proven to be useful, many initiatives can be taken to make improvements. Further, as the project was time-boxed, much considered functionality was not implemented due to lack of time. Additionally, several aspects discovered during the usability test could be addressed in order to enhance DigiPECS. In this chapter a short discussion of ways to improve and enlarge DigiPECS will take place.

Modifying pictures Much effort could be put on making DigiPECS more modifiable from a users perspective. From our collaboration partner Birken it was ascertained that through the use of a program on the computer, *BoardMaker*, the pedagogues could adjust pictures to each individual child and its needs. This meant changing size of the image, text and use of colours. Some children like colours, some do not. And the more skills a child have in the use of PECS, the smaller the image gets and the larger the text gets. This feature could improve DigiPECS considerably, making it much easier for the guardian to assist and teach the child communication and reading.

Import pictures As DigiPECS is now, it is a cumbersome task to transfer pictures from a computer. Pictures have to be stored on a SD memory card before an import to DigiPECS can take place. Two different solutions could improve this considerably. First, integration with BoardMaker would be ideal, as the pedagogues in Birken are used to working with this program. A second option could be to have a website, that from a computer, could allow a guardian to control the library on the unit. This would increase the efficiency considerably.

Printing of pictures A user story identified, but not implemented, is the possibility of connecting the tablet directly to a printer, making it possible to print pictures from DigiPECS directly. This would be a good feature for a guardian of a child, as certain situations still demand that PECS is used in its analogue form, e.g. a trip to the public swimming pool, the beach, funfairs etc.

Supports of multiple orientation modes The DigiPECS application has locked the direction in portrait orientation, and thereby restricted the length of the sentence board.

By adding a setting in the administration mode, making it possible to flip the tablet to landscape mode, construction of longer sentences could be possible. In doing so, the target group of DigiPECS could be widened to encompass children, or even adults, capable of constructing complex sentences.

Organisation of pictures in archive Both test persons from the usability test of the administration mode, requested that a way to organise pictures in the archive would be nice. Both were annoyed with the fact that the pictures were not sorted alphabetically or in some other way to make searching for a particular picture faster. One way to approach this could simply be to demand that all pictures had a text attached to it, used for organising the pictures alphabetically. Then, a setting could make it possible to hide/show the text of the picture in the DigiPECS book for the child. Another way to organise the pictures is to sort pictures in the archive into different categories, e.g. *Food*, *Toy*, *At grandparentâĂŹs house* etc. This functionality was identified in a user story, but due to time it was not implemented.

Warnings During the usability test of the administration mode, it was identified that deletion of pictures from the archive as well as removing locked pictures in the sentence board, happened without giving any warnings. This is a problem that could be solved easily by providing user feedback, asking if deletion is what (s)he wants. In this way accidental deletions are avoided and the usability is improved.

Mute button . In the usability test conducted on the children in Birken the guardians commented that sounds is not always desirable in a situation. It was requested that these sounds could be muted.

Camera implementation requires improvements Through the usability test several issues were found regarding the interface for taking pictures using the integrated camera. The interface is not intuitive. The orientation is locked to landscape mode, resulting in confusion for the user. The whole camera interface needs a rework to take these problems into consideration.

14.2 Future Works for GIRAF

Above we presented the conclusion for GIRAF. As stated, our tests showed that we were somewhat successful but several enhancements could be made. Some of these are presented here:

Access to administration mode During the usability test of DigiPECS administration mode, a side affect was that the key combination the guardian should press for entering the administration mode was tested. It was found that one of the test persons was annoyed by having to press the volume buttons and found this solution messy. The problem found was reported as an issue to the group responsible. An enhancement of accessing the administration mode could be to use a password. By pressing the wrong password, the user could be brought back to where s(he) was before entering the password interface. This interface could, to improve the security of it, be activated by pressing two of the touch units buttons at the same time. To ensure that a child does not get stuck in a password interface, a timeout could be started, making sure that the child would return to where (s)he was before, after a certain amount of time.

Make the launcher more customisable We believe, the launcher could be enhanced to be more personalise. The size and type of icons should be changable as so should background image.

BIBLIOGRAPHY

[AB10] Sony Ericsson Mobile Communications AB. Sony ericsson. http://www.sonyericsson.com, 2010. [Online - accessed 02-05-2011]. 35

[BCK03] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, 2nd edition, 2003. i, 47

[Ben10] David Benyon. *Designing Interactive Systems*. Pearson Educational Limited, 2nd edition, 2010. 42, 60

[BF94a] Andrew S. Bondy and Lori A. Frost. *PECS Håndbog - Kommunikationssytem med udveksling af billeder*. Pyramid Educational Products, Inc., 1994. 17

[BF94b] Andrew S. Bondy and Lori A. Frost. The picture exchange communication system. *Focus on Autistic Behaviour*, Vol. 9, 1994. 15, 16, 17

[BF99] Andrew S. Bondy and Lori A. Frost. *An introduction to PECS - The Picture Exchange Communication System*. Pyramid Educational Products, Inc., 1999. [DVD]. 16, 17, 18

[Coh04] Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison Wesley, March 2004. 23, 41, 42, 44

[Com11a] The Nielsen Company. Who is winning the u.s. smartphone battle? 2011. 35

[Com11b] Computerworld. Gartner: Android and apple win big globally in q1. http://www.computerworld.com/s/article/9216848/Gartner_Android_and_Apple_win_big_globally_in_Q1, 2011. [Online - accessed 23-05-2011]. 35

[Día11] Luis Cañas Díaz. Brief study of the android community. 2011. 35

[fBIN10] National Center for Biotechnology Information (NCBI). Autism. http://www.ncbi.nlm.nih.gov/pubmedhealth/PMH0002494/, 2010. [Online - accessed 20-05-2011]. 15

[Gar09] Vahid Garous. Requirements-based test generation. http://www.softqual.ucalgary.ca/projects/testing_labs/lab2/Lab%202%20-%20Requirements-Based%20Test%20Generation.pdf, 2009. [Online - accessed 02-05-2011]. 85

[Goo11] Google. What is android? http://developer.android.com/guide/basics/what-is-android.html, 2011. [Online - accessed 23-05-2011]. 38

[Hwa11] Hwaci. About sqlite. http://www.sqlite.org/about.html, 2011. [Online - accessed 06-03-2011]. 72

[Inc10] Google Inc. Android 2.2 compatibility definition. 2010. 37

[Inc11a] Google Inc. Android developers - home. http://developer.android.com/index.html, 2011. [Online - accessed 10-02-2011]. 8

[Inc11b] Google Inc. Android developers - package index. http://developer.android.com/reference/packages.html, 2011. [Online - accessed 10-02-2011]. 72

[Inc11c] Google Inc. Android developers - the developer's guide. http://developer.android.com/guide/index.html, 2011. [Online - accessed 10-02-2011]. 72, 85

[Inc11d] Google Inc. Android open source project. http://source.android.com/, 2011. [Online - accessed 02-05-2011]. 35

[JIS+06] Charlotte Holmer Jørgensen, Torben Isager, Bent Vandborg Sørensen, Lennart Pedersen, Bo Hejlskov Jørgensen, Birgitte Bjørn Jensen, and Marie Herholdt Jørgensen. National autisme plan. http://www.autismeportalen.dk/files/National%20Autisme%20Plan/NAP_Opsporing_web.pdf, 2006. 16

[JKS04] Mikael B. Skov Jesper Kjeldskov and Jan Stage. Instant data analysis: Conducting usability evaluations in a day. 2004. 78

[Kom10] Aalborg Kommune. Birken. http://www.aalborgkommune.dk/Borger/boern-og-unge/Handicap/institutioner/Specialboernehaver/Documents/Specialboernehaven-Birken-service.pdf, 2010. [Online - accessed 29-04-2011]. 19

[Lar03] Graig Larman. *Agile & Iterative Development: A Manager's Guide*. Addison Wesley, August 2003. 31, 41

[Mei10] Reto Meier. *Professional Android 2 Application Development*. Wiley Publishing, Inc., 2010. 38, 39, 72, 73

[MMMNS01] Lars Mathiassen, Andreas Munk-Madsen, Peter A. Nielsen, and Jan Stage. *Objektorienteret analyse & design*. Forlaget Marko ApS, Aalborg, 3. edition edition, 2001. 53

[Pat05] Ron Patton. *Software Testing*. Addison Publishing, July 2005. 85

[Pre10] Roger Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, USA, 7 edition, 2010. 88

[Rub07] Andy Rubin. Where's my gphone. 2007. 35

[SAM11] SAMSUNG. Samsung. http://samsung.com, 2011. [Online - accessed 02-05-2011]. 35

[Sch97] Ken Schwaber. Scrum development proces. 1997. 31

[SKS06] Abraham Silberschatz, Henry Korth, and S. Sudarshan. *Database Systems Concepts*. McGraw-Hill, Inc., New York, NY, USA, 5 edition, 2006. 72

[Sør99] Ester Ulsted Sørensen. Små børn med autisme - tidlig indsats. http://www.autisme.dk/filer/pjecer-til-download/sma-born-med-autisme-tidlig-indsats, 1999. 16

[VUBK09] Anna Voss, Dorte Haubro Uhrbrand, Jette Bach, and Emmy Kjelmann. Naar dit barn ikke taler. http://isaac.dk/downloads/naar_dit_barn_ikke_taler.pdf, 2009. 16

[Wat11] Gray Watson. Ormlite - lightweight object relational mapping (orm) java package. http://ormlite.com/, 2011. [Online - accessed 27-04-2011]. 72

[WwJTO09] Sven Windahl and Benno Signitzer with Jean T. Olson. *Using Communication Theory: An Introduction to Planned Communication*. Sage Publications Ltd, 2nd edition, 2009. 16

Appendices

APPENDIX A

USER STORIES

1: Move pictures around in sentence board At the meeting in Birken, it was stated that if a child placed pictures in the wrong order when making a sentence, the communication partner moves the picture in the correct order to teach the child. Further, the child should be given the opportunity to correct the order itself, if (s)he realises the order is incorrect.

User Story *As a child I want to move pictures around in the sentence board, so that pictures can be rearranged*

Notes Implement functionality to delete picture

Tasks to be implemented

- Add drag and drop in the sentence board
- Add drag and drop to delete pictures
- Add visual appearance when a picture is selected
- Add button in user interface to clear sentence board

Features to be tested :

- Swapping of pictures, when one picture is dragged to another position in sentence board
- Moving of pictures into empty spaces in sentence board.

2: Child starts DigiPECS This user story is the first to be implemented as it is the entry point to DigiPECS. It should be started from the launcher where an icon representing DigiPECS is available, and should bring the child to the personalised DigiPECS book. As DigiPECS most likely will be accessed several times throughout the day, is should be fast to access and the behaviour should be predictable. When the program is started, the child's own personalised DigiPECS book is shown.

User Story *As a Child I want to start the program, so that I can use it for communication*

Notes Needs integration with other sub-projects.

Tasks to be implemented

- Create Eclipse project
- Create an extended image view, containing a picture and text

- Integrate multi project libraries and test them
- Set up Doxygen documentation

Features to be tested :

- Start up of DigiPECS when icon in launcher is clicked

3: Move picture into sentence board The main focus with this user story is to ensure a child can move pictures from the DigiPECS book to the sentence board. The sentence board is closely related to the sentence strip found in the physical PECS book. This can be achieved in multiple ways, but through the meeting with Birken it was determined that using drag and drop functionality would be a good solution. Drag and drop, closely imitates the actual physical hand movement of taking a picture and move it into a sentence strip found in the PECS book.

User story *As a Child I want to move pictures into the sentence board, so that I can create sentences*

Notes Drag and drop, define contents in a picture

Tasks to be implemented

- Create sentence board layout
- Add drag and drop features from book to sentence board
- Create gridview, in sentence board, for holding images

Features to be tested :

- Drag and drop
- Sentence board

4: Change category *NOT IMPLEMENTED*

User story *As a Child I want to change category, so that I can find the picture I need*

Notes Needs database design (E-R diagram)

5: Record a voice pronunciation As one of the purposes of PECS is to teach the child (oral) communication skills, this feature is implemented. This allows for letting a pronunciation of a word associated with a picture be available whenever new pictures are added to DigiPECS. Further, by letting the guardian (or the child, with help from the guardian) record a voice pronunciation, the child is familiar with the voice reading up the word associated with the picture chosen.

User story *As a guardian I want to record a pronunciation of the word, so the child can hear the spoken word and learn the relation between the picture and the real object*

Notes Child's voice can be used

Tasks to be implemented

- Modify database and model layer to support addition of sound
- Create a media recorder implementing functionality for playing and capturing voices
- Modify the user-interface with buttons for recording and playing sound

Features to be tested :

- Audio recording
- Play when picture is touched in sentence board and book

6: Sort pictures into categories *NOT IMPLEMENTED*

User story *As a guardian I want to sort pictures in categories, so that I can arrange pictures to our (child's and my) needs*

Notes May be integrated with admin sub-project

7: Activate pictures As a result from the meeting in Birken, a need emerged to be able to control which pictures were available for the child. In the physical PECS book a picture is taken out of the book when the item is not available. This should be mapped into DigiPECS in an efficient way, making in possible to quickly activate/deactivate pictures shown to the child.

User story *As a guardian I want to control what pictures are active, so that I can restrict what the child may get/do*

Notes – Decide if a a picture is deactivated for a situation only or for the whole PECS book.

Tasks to be implemented

- Update database to include support for activating/deactivating pictures
- Update the model to include support for activating/deactivating pictures
- Create an administration interface for activating and deactivating pictures

Features to be tested :

- Activation/deactivation in administration mode
- Available/unavailable in book

8: Change page Contrary to the physical PECS book, DigiPECS can hold an almost arbitrary number of pictures. Often, more pictures will be present in the DigiPECS book than can be shown in one page. To map the physical counterpart of changing page in a book, an intelligent way to handle this situation has to be devised.

User story *As a Child I want to change pages, so that I can have more pictures than one screen allows*

Notes How to change page (slide/button)? Generically create number of pages. How to handle 20 pages?

Tasks to be implemented

- Create page line
- Implement page-change in gridview
- Adjust page size in admininistration mode.

Features to be tested :

- Page line
- Page change
- Correct pictures in each page

9: Rearrangement of pictures From the collaboration meeting at Birken it was experienced that some of the children like to arrange the pictures in their own PECS book. The children make their own system and moves the PECS pictures to the pages they like and in the order they like independently of the pedagogue. This feature should also be available in DigiPECS, and has to be implemented in a way that is intuitive for the child to rearrange the pictures.

User story *As a child I want to rearrange pictures on a page, so that I can personalise my pages*

Notes It should be possible to rearrange pictures on each individual page - but also to move a picture to another page. How do pictures get swapped logically?

Tasks to be implemented

- Add drag and drop functionality on each page
- Add drag and drop functionality to move pictures from one page to another

Features to be tested :

- Swapping when one picture is moved to a postition with another picture
- Same across pages

10: Sentence read out loud Even though PECS primarily is used by people with no oral communication skills, a positive side effect of using PECS is that some of the children can learn to speak or at least understand simple words or sentences, see section 3.3. Further, it is a steady procedure in Birken that whenever a child places PECS pictures in the sentence strip, the pedagogue reads aloud the word(s) associated with the picture. Because of this, DigiPECS should be able to read aloud sentences, so the child continuously can associate objects and desires with sounds.

User story *As a child I want to have sentences read out loud, so that I can hear how to pronounce them*

Notes Only the sound associated with the pictures should be read out loud – not the gender.

Tasks to be implemented

- Add button to the sentence board layout
- Implement a sentence function to play a sequence of audio files

Features to be tested :

- Sentence board audio

11: Pictures read out loud Even though PECS primarily is used by people with no oral communication skills, a positive side effect of using PECS is that some of the children can learn to speak or at least understand simple words or sentences, see section 3.3. Further, it is a steady procedure in Birken that whenever a child places PECS a picture in the sentence strip, the pedagogue reads aloud the word associated with the picture. Because of this, DigiPECS should be able to read aloud words associated with a picture, so the child continuously can associate objects and desires with sounds.

User story *As a child I want to have pictures read out loud when I touch them, so that I can hear how to pronounce them*

Tasks to be implemented

- Refactoring the database and model layer to support sound
- Implement audio player in the PECS book to play sound

Features to be tested :

- Audio with pictures
- Not queueing

12: Set up situations During the meeting in Birken it was discovered that before breakfast, lunch and other situations, each individual PECS book was updated to match the available items during that particular situation. To make this more efficient in DigiPECS, it should be possible to set up different situations, and chose one of them to be visible to the child. This would make the process of setting up a situation faster, as not all pictures has to be updated in each new situation.

User story *As a guardian, I want to set up situations, so that I can restrict the available pictures to the situation without the need to do it manually*

Tasks to be implemented

- Create tables in, and modify, the database to support situations.
- Modify the model layer to support situations
- Design and implement a user interface for selecting an active situation
- Design and implement a user interface for creating and removing situations.
- Design and implement a user interface for adding and removing pictures to/from situations

Features to be tested :

- Situations in administration mode
- Available in book in accordance with administration

13: Administration mode As the administration mode should be inaccessible to the child, a clear way of identifying when administration mode is accessed is needed. This way, a guardian positioned at visual distance from the tablet can easily recognise if the child accidentally has gained access. A black colour-theme only used in administration mode can be used for this.

User story *As a guardian I want to see when the child is in administration mode, so that I can prevent the child from changing settings*

Notes Black background. Not our responsibility.

Features to be tested :

- Colour in book and administration different
- Accordance with 'Administration'-group's interface

14: Lock picture to position in sentence board In the collaboration meeting with Birken a specific feature request was stated. When using PECS a picture is often placed permanent in the sentence board helping the child learning to create sentences. Often this picture is 'I want'. From Birken it was stated that it would be good if there could be a way to lock a picture in the sentence board.

User story *As a guardian I want to be able to lock a picture to a position in the sentence board, so that I can teach the child about the present of a "I want" picture when constructing sentences.*

Tasks to be implemented

- Refactoring the database and model layer to support locking of tasks
- Create a simple administration interface to allow guardians to lock and unlock pictures.

Features to be tested :

- Locked pictures may no be dragged from the Sentence board
- Locking picture in administration mode

15: Adding picture with camera From the Birken collaboration meeting we realised that one of the time consuming tasks are the one creating and adding customised pictures to a PECS book. Pictures have to be taken using a digital camera and edited and printed on a computer before they can be physically laminated. In DigiPECS the integrated camera can be utilised to take, edit and share pictures in one step. This is a reduction in the workload of having to create new pictures, hopefully allowing a more customised book.

User story *As a Guardian I want to take pictures with the camera, so that I can easily add a picture of the item in front of me*

Notes User interface needs to be designed and implemented

Tasks to be implemented

- Refactor code to optimise re-usability of components
- Integrate with the Android camera interface to capture pictures
- Add picture taken with the camera to the PECS book

Features to be tested :

- Camera in administration mode
- DigiPECS book

16: Adding pictures As with user story 15, the incentive for this user story is to reduce workload of creating new pictures and make a more customised book. Therefore, in addition to adding pictures taken by the integrated camera, it should also be possible to add pictures to the DigiPECS book that have already been saved on the device.

User story *As a guardian, I want to add new pictures to the PECS book, so that I can extend the selections for the child*

Notes Set up activity for admin interface

Tasks to be implemented

- Refactor database design

- Change resources to files
- Import and present pictures from Android Media Store Library
- Implement adding from Media Store to the PECS book

Features to be tested :

- Addition of pictures in the administration
- DigiPECS book

17: 'Help me' button *NOT IMPLEMENTED*

User story *As a child, I want a 'help me' button available at all time, so that I can ask for help if I need it*

Notes A 'Help me' picture has been introduced instead

18: Modify pictures The administration interface should integrate functionality that allows the guardian to modify pictures by giving them names and recording sound tracks.

User story *As a guardian, I want to be able to modify a picture so that it fits the childŠs needs*

Notes Extend database and model layer to support modification

Tasks to be implemented

- Design a suitable user-interface for manipulating settings
- Add functionality for adding or modifying text to picture
- Add functionality for adding or modifying sound to picture

Features to be tested :

- Administration interface
- Changed settings in book

19: Start secured administration mode The administration interface should be inaccessible for the child. However, because administration of the DigiPECS book should be done continuously throughout the day by the guardian this interface should be easily accessible. How this is accessed is the responsibility of the group developing the launcher.

User story *As a guardian I want a separated and secured administration mode, so that the application settings is inaccessible for a child*

Notes Design paper prototype. Has to be integrated with administration library. Must be tested with GIRAF-launcher

Tasks To be implemented

- Create initial xml-file for admin library
- Create paper prototype for admin pages

Features to be tested :

- Administration mode can be lauched through the multi-project
- Administration mode

20: Print pictures *NOT IMPLEMENTED*

User story *As a guardian, I want to print pictures, so that I can use them as physically PECS pictures*

21: Drop target It should be possible to clearly identify the area, where an object may be dropped, when the drag is started.

User story *As a child, I want to have a visual change in DigiPECS when I select a picture, so I know where I can drag the picture to*

Notes Create drag-listener. Must be coordinated with colour-theme.

Tasks to be implemented

- Add visual appearance when a picture is selected

Features to be tested :

- Sentence Board - colours when dragging and when not dragging

22: Colour theme *NOT IMPLEMENTED*

User story *As a child, I want to change colour theme, so that I can get my favourite colours*

Notes Can be done with several colour.xml files

23: Deactivate sound *NOT IMPLEMENTED*

User story *I as a guardian want to deactivate sound*

APPENDIX B

USER STORY DIAGRAMS

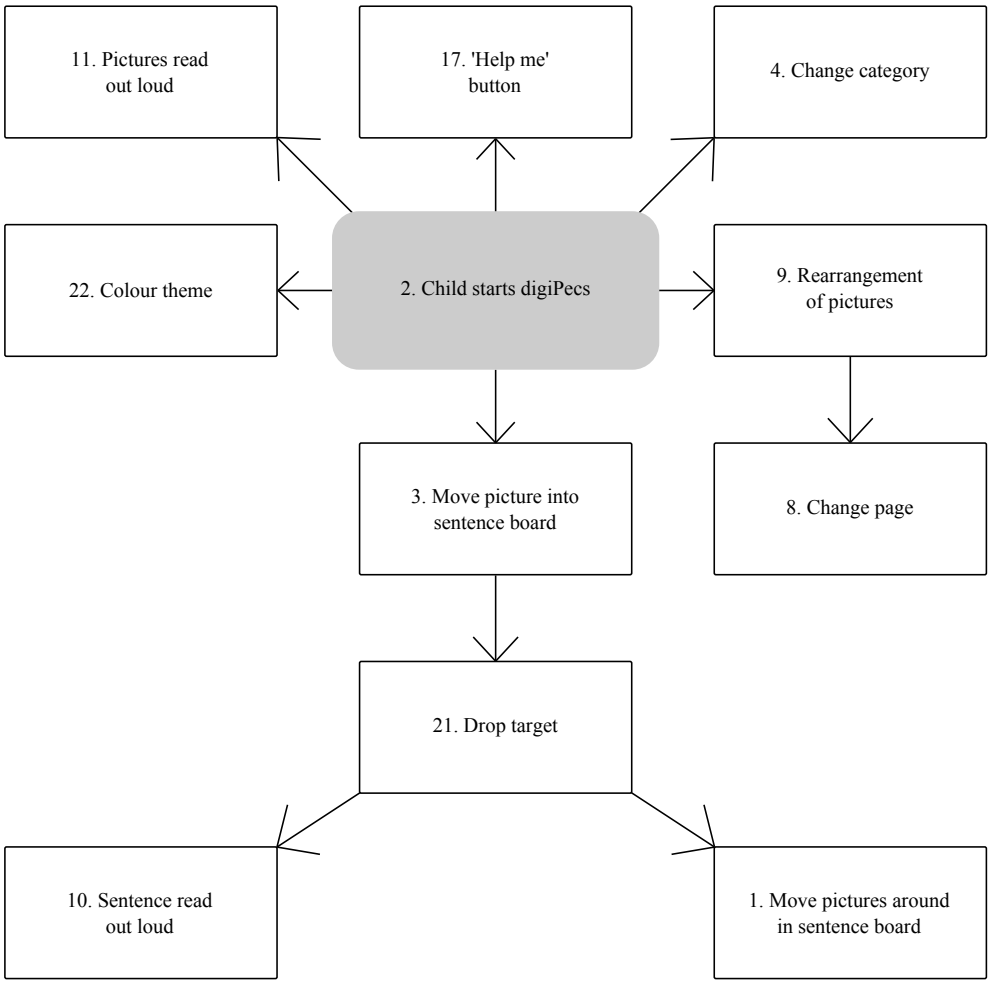


Figure B.1: User Story Diagram for the User Role **Child**

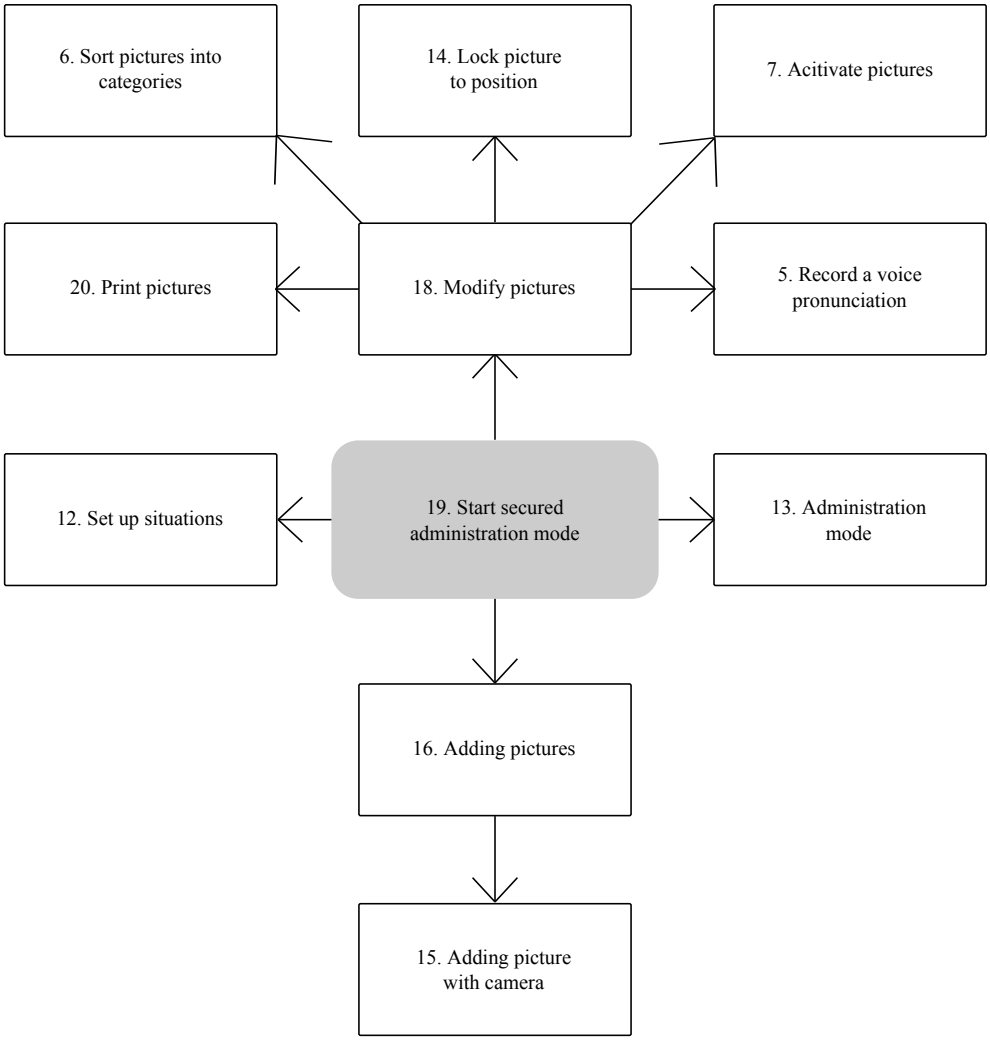


Figure B.2: User Story Diagram for the User Role **Guardian**

APPENDIX C

NAVIGATIONAL DIAGRAM

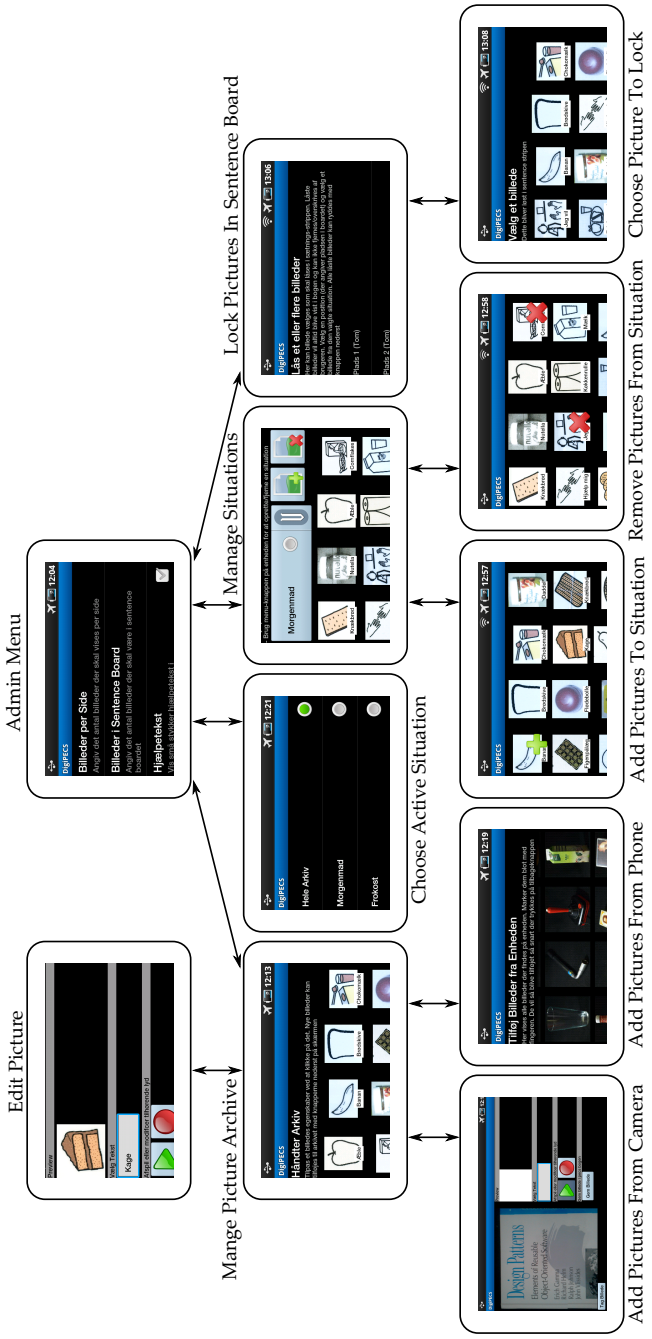


Figure C.1: A navigation diagram showing how a user might navigate through the administration interface

APPENDIX D

USABILITY TEST

Introduktion til DigiPECS

- **Kender brugeren til touch units?**
  - Knapper, Android, Touch, Drag & Drop
- **Kort intro til DigiPECS børne siden**:
  - Denne side er beregnet til børnene
  - Børnesiden viser én situation: f.eks. Morgenmad, legeaktiviteter
  - Der er mulighed for at skifte side, hvis der er flere billeder i situationen end der kan være på siden.
  - Børnene kan frit flytte rundt på billederne:
    - De kan ændre på rækkefølgen de vises i ved at bytte om på dem
    - De kan flytte billederne ned i sætningsfelterne
    - Billederne kan også frit flyttes rundt i sætningen og de kan fjernes derfra igen
- **Notation og principper**:
  - PECS Arkiv: Stor samling billeder, hvor man tilføjer de billeder man ønsker at have tilrådighed, når man sammensætter situationer.
  - Situation: En lille samling billeder der er tilpasset til en bestemt situation i barnets hverdag: f.eks. morgenmad. Billeder bliver tilføjet hertil fra arkivet.
  - "Hele arkiv" - én situation der viser alle billeder i arkivet.
- **Administrationssiderne**:
  - Hvordan kommer man ind i admin mode?
    - Volume op/ned/op/ned/op/ned
  - Tilføje billeder fra kamera eller telefon til arkivet
  - Bruges til at håndtere situationer
    - Tilføje/fjerne situation
    - Tilføje/fjerne billeder til en situation, fra arkivet
    - Et billede i én situation kan deaktiveres midlertidigt, f.eks. hvis mælken bliver tom under morgenmaden.
  - Låse billeder fast i sætningen, f.eks. "Jeg vil have" i første felt
  - Ændre antallet af billeder der vises per side og antallet af billeder i sætningen

Test opgave

Tak fordi du vil deltage i denne test af digiPECS. Vi vil gerne understrege at det udelukkende er DigiPECS vi tester for fejl og mangler og ikke dig. Det er meget vigtigt at du husker at tænke højt. Uden kritik er vi ikke i stand til at finde de fejl og mangler systemet har.

Opgave 1: Morgenmad

Du skal forestille dig at du er ved at forberede morgenmad i børnehaven. Et af børnene har en DigiPECS bog, som du er ved at gøre klar og tilpasse til de produkter der står på bordet i dag. Der er ikke mere smør, og du skal derfor have fjernet smør fra den situation der hedder morgenmad. Desuden skal du have tilføjet malk til situationen morgenmad. Malk er et billede som allerede ligger på enheden, men som ikke pt. er aktiv i morgenmad-situationen. Opgaven er som følger:

- Åben applikationen DigiPECS
- Åben administrationsdelen
- Deaktiver smør fra situationen morgenmad
- Tilføj malk til situationen morgenmad og aktiver denne situation
- Kontroller at alene morgenmad situationen er tilgængelig for barnet

Opgave 2: Legetøj

Morgenmaden er nu overstået, og det er tid til leg. I har fået et nyt stykke legetøj - en bamse - i børnehaven, og det er endnu ikke lagt ind i digiPECS. Opgaven er som følger:

- Tilføj et billede til arkivet ved hjælp af DigiPECS indbyggede kamera funktion
- Tilføj teksten "Bamse" til billedet af bamsen
- Indspil din egen udtale af ordet "Bamse" til billedet af bamsen
- Tilføj billedet af bamsen til situationen Vælgebog, som allerede eksisterer i DigiPECS
- Aktiver Vælgebog situationen, og kontroller herefter at det vises korrekt for barnet

Opgave 3: Frokost

Nu er det tid til frokost. Da det er første dag digiPECS bruges i børnehaven, er frokost situationen endnu ikke oprettet. Opgaven er som følger:

- Opret en ny situation Frokost
- Tilføj brød, ketchup og salat til situationen.
- Aktiver den nyoprettede Frokost situation.
- Billedet brød skal nu låses til sætningstippen i 2. position. Kontrollere at dette billede er fastlåst i den rette position i digiPECS, og at alene brød, ketchup og salat vises for barnet

TAK FOR DIN DELTAGELSE

APPENDIX E

UNIT TEST

This is lists `DataBaseSituationTableTEST` class with all appertaining test cases.

```
1  package sw6.digipecs.test;
2
3  import sw6.digipecs.R;
4  import sw6.digipecs.database.DBAdapter;
5  import sw6.digipecs.database.SituationCursor;
6  import sw6.digipecs.exceptions.DatabaseException;
7  import android.content.Context;
8  import android.test.AndroidTestCase;
9  import android.util.Log;
10 /**
11  * Will test operations on Situation table works as expected. Reliant on
12  * fetchsituations and fetchsituation are implemented correct.
13  * @author sw6.digipecs
14  */
15 public class DataBaseSituationTableTEST extends AndroidTestCase {
16     private final static String LOG_TEXT = "sw6.digipecs.test";
17     private DBAdapter mDb;
18     private String mName;
19     private long mRowID01;
20     private long mRowID02;
21     private long mRowID03;
22     private Context mContext;
23
24     @Override
25     protected void setUp() throws Exception {
26         super.setUp();
27         //Initialise variables
28         mName = "TestName";
29         mContext = getContext();
30         mDb = new DBAdapter(mContext);
31         //Setup test environment
32         mDb.open();
33         mRowID01 = mDb.addSituation(mName + 1);
34         mRowID02 = mDb.addSituation(mName + 2);
35         mRowID03 = mDb.addSituation(mName + 3);
36     }
37
38     @Override
39     protected void tearDown() throws Exception {
40     super.tearDown();
41         //Teardown test environment
42         mDb.deleteSituation((int)mRowID01);
43         mDb.deleteSituation((int)mRowID02);
44         mDb.deleteSituation((int)mRowID03);
45         mDb.close();
46         //reset variables
47         mContext = null;
48         mName = null;
49     }
50
51     public void testQueryAll(){
52         //Query database for all situations in the situation table
```

```
53          SituationCursor sCursor =  mDb.fetchSituations();
54          //expected number of rows - only works on a clean database
55          final int expectedNoRows = 4;
56          assertTrue("01 test fetch situation ",  sCursor instanceof SituationCursor );
57          assertEquals("02 test numbers of rows", expectedNoRows, sCursor.getCount());
58          //Clean Up
59          sCursor.close();
60      }
61
62      public void testQuery(){
63          //Get a situation cursor with one row
64          SituationCursor sCursor =  mDb.fetchSituation(0);
65          //Expected number of rows
66          final int expectedNo = 1;
67          assertTrue("01 test fecth situation ",  sCursor instanceof SituationCursor );
68          assertEquals("01 test numbers ", expectedNo, sCursor.getCount());
69
70          //Clean Up
71          sCursor.close();
72      }
73
74      public void testAddSitation() throws DatabaseException{
75          //Get a situation cursor with all rows
76          SituationCursor sCursor = getSituationCursor();
77          //Get the number of rows in this cursor
78          final int countBefore = sCursor.getCount();
79          //Add a new situation
80          final int rowID = (int)mDb.addSituation(mName + 1);
81          //Update the cursor
82          sCursor = getSituationCursor();
83          //Get the count after
84          int countAfter = sCursor.getCount();
85
86          assertTrue("01 test add situation ", countAfter - countBefore == 1);
87          //CLEAN UP
88          mDb.deleteSituation(rowID);
89          sCursor.close();
90      }
91
92      public void testDeleteSituation() throws DatabaseException{
93          final int rowID = (int)mDb.addSituation(mName + 1);
94          SituationCursor sCursor = getSituationCursor();
95          final int countBefore = sCursor.getCount();
96
97          mDb.deleteSituation((int)rowID);
98          sCursor = getSituationCursor();
99          final int countAfter = sCursor.getCount();
100
101         assertEquals("01 test add situation on situation table ", 1, countBefore - countAfter);
102
103         //Clean Up
104         sCursor.close();
105     }
106
107     public void testgetSituationAttributes(){
108         //Default situation created when database is created
109         final int expectedID = 0;
110         final boolean expectedActive = true;
111         //Get the name of the situation from the resources in the application
112         final String expectedName = mContext.getString(R.string.defaultSituationNameDB);
113         //Get a cursor pointing to this situation
114         SituationCursor sitCursor = getSituation(expectedID);
115         //Test getter of this cursor
116         assertEquals("01 test get situation name ", expectedName, sitCursor.getName());
117         assertEquals("02 test get situation id ", expectedID, sitCursor.getId());
118         assertEquals("03 test get situation active ", expectedActive, sitCursor.getActive());
119         //Clean Up
120         sitCursor.close();
121     }
122
123     public void testsetSituationName(){
124         SituationCursor sCursor = getSituationCursor();
125         //Gets last situation found in the SituationCursor
126         sCursor.moveToLast();
127         //Gets the id of the situation
128         final int sid = sCursor.getId();
129         //Create a new name for the situation
130         final String newName = "NewName";
131         mDb.setSituationName(sid, newName);
132
```

```
133         //Gets an SitiationCursor that returns one situation based on the id
134         SituationCursor sitCursor = getSituation(sid);
135 
136         //Just to ensure we are still working on the same situation ID
137         assertEquals("01 test get situation id ", sid, sitCursor.getId());
138         //Checks to see if the name has been changed to the newly added name
139         assertEquals("02 test get situation name ", newName, sitCursor.getName());
140 
141         //Clean Up
142         sCursor.close();
143         sitCursor.close();
144     }
145 
146     public void testsetAndGetSituationActive() throws DatabaseException{
147          SituationCursor sCursor = getSituationCursor();
148          //Gets last situation found in the SituationCursor
149          sCursor.moveToLast();
150         //Gets the id of the situation
151         final int sid = sCursor.getId();
152         //Create a new name for the situation
153         final boolean activeState = true;
154         mDb.setSituationActive(sid);
155 
156         //Gets an SitiationCursor that returns one situation based on the id
157         SituationCursor sitCursor = getSituation(sid);
158 
159         //Just to ensure we are still working on the same situation ID
160         assertEquals("01 test get situation  id ", sid, sitCursor.getId());
161         //Checks to see if the name has been changed to the newly added name
162         assertEquals("02 test get situation active ", activeState, sitCursor.getActive());
163         //Checks get active state
164         assertEquals("03 test get situation that active on db based on id", sid, mDb.↩
                getSituationActive());
165         //Clean Up
166         mDb.setSituationActive(0);
167         sCursor.close();
168         sitCursor.close();
169      }
170 
171      //Private methods to fetch cursors with.
172     private SituationCursor getSituationCursor(){
173         SituationCursor rtnCursor = mDb.fetchSituations();
174         rtnCursor.moveToFirst();
175         return rtnCursor;
176     }
177     private SituationCursor getSituation(int sid){
178         SituationCursor rtnCursor = mDb.fetchSituation(sid);
179         rtnCursor.moveToFirst();
180         return rtnCursor;
181     }
182 }
```

APPENDIX F

MULTI PROJECT INTEGRATION & SYSTEM TEST

Test Plan

Tested Versions:

All tests are conducted on the following dev-branches, revision 1122:
sw6.admin: /sw6.admin/branches/dev/
sw6.bmi:/sw6.bmi/branches/dev/
sw6.lib: /sw6.schedule/branches/dev/
sw6.schedule: /s6w.schedule/branches/prototype1

Notice that sw6.pecs contained errors in revision 1122, hence the source is not included in this test. A stable apk is available, and will be used when possible.

Identifier: TD00

Program/Module/Object Under Test:

GirafPlace application administration panel, web interface.

Test Objective:

Ensuring GIRAF applications can be deployed to GIRAF Place, and that non-GIRAF applications and corrupted GIRAF applications are rejected.

Test Conditions and Intercase Dependencies:

The source must be available for all tested applications. The PECS source is not buildable at the time of this test execution, hence steps requiring the source has been skipped for this application.

Test Procedure:

Instruction	**Expected Behaviour**	**Pass / Fail**
Create an Android application with an invalid settings.xml (e.g. verify that it is invalid using the sw6.xmlvalidator tool)	The settings.xml is considered invalid by the sw6.xmlvalidator tool.	**aSchedule**: Pass **bmi**: Pass
Upload the application to GIRAF Place.	The application is rejected by the GIRAF Place, and should report the same errors as the sw6.xmlvalidator tool.	**aSchedule**: Fail (application is accepted) **bmi:** Fail (application is accepted)
Correct the settings.xml file, and upload the application to GIRAF Place again.	The application is accepted and stored on the GIRAF Place.	**aSchedule:** Pass **bmi:** Pass
Verify that the settings.xml file is valid using the sw6.xmlvalidator tool	sw6.xmlvalidator reports file is valid	**aSchedule:** Pass **bmi:** Pass
Update the application with a new	The application will be upgraded, and stored	**aSchedule:** Pass

change in its code. Remember to edit the version number. Upload the updated application to GIRAF Place.	on GIRAF Place.	**bmi:** Pass

Test Results and Observations:

Applications with invalid settings.xml are accepted at GIRAF Place, this was not intended. Upgrade of packages worked as intended. The sw6.xmlvalidator tool reported the correct problems introduced in the settings.xml files.

Identifier: TD01

Program/Module/Object Under Test:

GIRAF.

Test Objective:

Testing initial installation of GIRAF, and that the user is presented with necessary welcome dialogues.

Test Conditions and Intercase Dependencies:

The latest revision (r1122) of GIRAF should be available at: http://girafplace.lcdev.dk/start/

Test Procedure:

Instruction	**Expected Behaviour**	**Pass / Fail**
Download sw6.giraf from website (http://girafplace.lcdev.dk/start/)	The apk file is downloaded.	**Desire:** pass **Hero:** pass **Wildfire:** pass **Tab:** pass
Install apk file.	GIRAF is installed.	**Desire:** pass **Hero**: pass **Wildfire:** pass **Tab:** pass
Set GIRAF as the default launcher.	GIRAF Launcher is started. A first run dialog should appear.	**Desire:** pass **Hero:** pass **Wildfire**: pass **Tab:** pass
Setup user profile - Name: John - Address: Doe Street 1337 - Parent's Phone: +45 1234 5678 - Gender: boy - Birthday: 01-02-2003 - Capabilities (mark following): - Can drag-and-drop	The user profile setup is stored.	**Desire:** pass **Wildfire:** pass **Hero**: pass **Tab:** pass

- Can read - Can Work with Numbers		

Observations

All tests can be passed. The following issues were discovered:

Rotating the user profile screen clears text fields:
- Open User Profile. Edit name. Close dialog. Edit name. Rotate screen. Name field is now empty

Cursor not placed at the end:
- Open a text dialog, e.g. “Name”. Enter the name. Close the dialog. Open the dialog again, and now the cursor is placed at the beginning, and cannot be moved to the end. This issue is observed on all HTC devices running HTC Sense UI.

Admin loads slowly
On HTC wildfire, the administration module loads slowly.

Identifier: TD02

Program/Module/Object Under Test:

Launcher, GIRAF place, applications including “in-app” administration.

Test Objective:

To ensure applications can be installed on device using GIRAF place. To ensure that the launcher and applications are able to use and store settings in Lib, as well as abiding to the navigation flow for opening in-app administration mode that has been agreed upon.

Test Conditions and Intercase Dependencies:

TD01.

Test Procedure:

Instruction	**Expected Behaviour**	**Pass / Fail**
Start GIRAF Place.	Ensure that apps are listed and filtered according to user profile (only apps filtered according to user profile should be possible to download).	**Tab (aSchedule):** Pass **Desire (bmi):** Pass **Hero(pecs, sched)**: Pass **Wildfire(digiPecs):** Pass
Download and install an application	The download should start. After download has finished, verify that the Android Package Manager installer starts, and that the installation succeeds.	**Tab (aSchedule):** Passb **Desire (bmi):** Pass **Hero(pecs, sched)**: Pass **Wildfire(digiPecs):** Pass
Navigate to launcher screen.	The download should start. After download	**Tab (aSchedule):** Pass

Check that only GIRAF applications are shown	has finished, verify that the Android Package Manager installer starts, and that the installation succeeds.	**Desire (bmi):** Pass **Hero(pecs, sched)**: Pass **Wildfire(digiPecs):** Pass
Launch an application	Verify that the top panel is not shown.	**Desire (bmi):** Pass **Tab (aSchedule):** Pass **Hero(pecs, sched)**: Pass **Wildfire(digiPecs):** Pass
Go into admin mode (using "secret" key combination)	Verify that the admin settings is the admin settings for the particular app being tested.	**Desire (bmi):** Pass **Tab (aSchedule):** Pass **Hero (pecs, sched):** pass **Wildfire:(digiPecs):** Pass
Change a setting for the application	Setting is updated.	**Desire (bmi):** Pass **Tab (aSchedule):** Pass **Hero (pecs, sched)**: Pass **Wildfire(digiPecs):** Pass
Press back until home screen is shown.	Verify that the activities are shown in the following order: Admin panel, App, Home screen.	**Desire (bmi):** Pass **Tab:** Pass **Hero (pecs, sced):** pass **Wildfire(digiPecs):** Fail
Start the app again	App opens.	**Desire (bmi):** Pass **Tab (aSchedule):** Pass **Hero (pecs, sched)**: pass **Wildfire(digiPecs):** Fail
Go into admin mode	Verify that the changes made before are the ones shown.	**Desire (bmi):** Pass **Tab (aSchedule):** Pass **Hero (pecs, sched)**: pass **Wildfire(digiPecs)**: Pass

Test Results and Observations:

Applications can be installed using GIRAF place. This makes them visible in the launcher and in the administration panel. If settings are stored using the “in-app” administration mode, they are stored persistently as expected.

- Press back until home screen is shown-failed with digiPecs: (running with old lib and bug is fixed) digiPecs needs to update newest lib-library.

- Refresh context menu in Giraf place cannot be read on HTC Wildfire. Icon is toooo big.

Identifier: TD03

Program/Module/Object Under Test:

GirafPlace client (application installer and updater), as well as launcher application display (listing of applications).

Test Objective:

Installing and upgrading GIRAF and GIRAF applications.

Test Conditions and Intercase Dependencies:

Requires TD02 to be executed.

Test Procedure:

Instruction	Expected Behaviour	Pass / Fail
Open GIRAF Place client (and ensure that an upgraded version of the package installed before, has now been uploaded to GIRAF Place. The upgraded version should introduce some changes in the settings.xml file)	The app installed earlier in this test, should be able to be upgraded.	**Tab (bmi):** Pass* **Desire (bmi):** Pass* **Hero (pecs)**: Pass* **Wildfire(digiPecs, bmi):** Pass*
Download the upgraded version.	The new version is downloaded.	**Tab (bmi):** Pass** **Desire (bmi):** Pass** **Hero (pecs)**: Pass*** **Wildfire(digiPecs, bmi):** Pass
Upgrade the downloaded application.	The Android Package Manager installer opens, and starts a “package replacement”.	**Tab (bmi):** Pass **Desire(bmi):** Pass **Hero (pecs):** Pass **Wildfire(digiPecs, bmi):** Pass
Go to launcher.	Updated application must be shown correctly.	**Tab (bmi):** Fail*** **Desire(bmi):** Fail*** **Hero (pecs):** Fail*** **Wildfire(digiPecs, bmi):** Pass***
Launch the application again.	Verify that the settings have been upgraded correctly by checking the application, and the admin mode.	**Tab (bmi):** Pass **Desire(bmi):** Pass **Hero (pecs):** Pass **Wildfire(digiPecs, bmi):**Pass

Test Results and Observations:

* If one forgets to update the GIRAF Place application listing, and the following assertions are true, then GIRAF Place is not able to install the wanted application.

- The application is not installed on the device.
- The application has been upgraded on GIRAF Place compared to the version shown on the list on the device.

** If one forgets to update the GIRAF Place application listing, and the application is currently installed on the phone, GIRAF Place shows an Uninstall button.

*** The application is now shown twice in the launcher. Both icons open the application though.
Install BMI4Kidz -> Verify that one icon is shown in the launcher -> Upload new version of BMI4Kidz to Market Place -> Upgrade the BMI4Kidz app -> go to the home screen -> now, two BMI4Kidz icons are visible on the home screen (Also an issue when updating PECS)

Icons in the sw6.admin, and sw6.bmi package (high res) does not follow the Android design guideline: http://developer.android.com/guide/practices/ui_guidelines/icon_design.html

Identifier TD04

Program/Module/Object Under Test:

Back Button behaviour, GIRAF launcher, and applications

Test Objective:

To test the back button behaviour of GIRAF client and apllications

Test Conditions and Intercase Dependencies:

TD01

Test Procedure:

Instruction	**Expected Behaviour**	**Pass / Fail**
Navigate to the home screen of GIRAF launcher and start Admin panel, by using secret key combination	Admin Panel is started	**Desire:** pass **Hero:** pass **Tab:** pass **Wildfire:** pass
Press the back button	GIRAF launcher home screen is shown	**Desire:** pass **Hero:** pass **Tab:** pass **Wildfire:** pass
Press the back button again (while still in the GIRAF launcher home screen)	Nothing will happen	**Desire:** pass **Hero:** pass **Tab:** pass **Wildfire:** pass
Open application with more than one activity	Application is started	**Hero (aSchedule):** pass **Tab (aSchedule):** pass **Desire (aSchedule):** pass **Wildfire(aSchedule)::**

		pass
Open secondary activity from the main activity	Secondary activity is shown	**Hero (aSchedule):** pass **Tab (aSchedule):** pass **Desire (aSchedule):** pass **Wildfire(aSchedule):** pass
Open admin panel from the application	Admin panel is started	**Hero (aSchedule):** pass **Tab (aSchedule):** pass **Desire (aSchedule):** pass **Wildfire(aSchedule):** pass
Press the back button(ensure that we currently are in the root of the Admin panel)	Secondary activity is shown	**Hero (aSchedule):** pass **Desire (aSchedule):** pass **Wildfire(aSchedule):** pass
Press the back button	Main activity is shown	**Hero (aSchedule):** pass **Tab (aSchedule):** pass **Desire (aSchedule):** pass **Wildfire(aSchedule):** pass
Press the back button	GIRAF launcher home screen is shown	**Hero (aSchedule):** pass **Tab (aSchedule):** pass **Desire (aSchedule):** pass **Wildfire(aSchedule):** pass
Press the back button	Nothing happens	**Hero (aSchedule):** pass **Tab (aSchedule):** pass **Desire (aSchedule):** pass **Wildfire(aSchedule):** pass

Test Results and Observations:

All step in this test passed and the back button behaves as expected.

Identifier:

TD 05

Program/Module/Object Under Test:

Behaviour of the home key.

Test Objective:

To test whether the home key handling works as intended.

Test Conditions and Intercase Dependencies:

TD01 is executed and passed. A valid application matching the user profile have been installed.

Test Procedure:

Instruction	**Expected Behaviour**	**Pass / Fail**
Navigate to home screen (GIRAF launcher)	see the GIRAF home screen.	**All:** pass
Press key combo (vol up - vol down) * 3	Admin panel opens	**All**: pass
Press home button	GIRAF home screen is shown	**All**: pass
Press home button	Nothing happens/GIRAF home screen is still shown	**Hero**: pass **Tab:** pass **Desire:** pass **Wildfire:** pass
Click an app to open it	The clicked app opens.	**Hero (aSchedule)**: pass **Tab (aSchedule):** pass **Desire (bmi):** pass **Wildfire:** pass
Press key combo (vol up - vol down) * 3	Admin panel opens	**Hero**: pass **Tab:** pass **Wildfire:** pass
Pres home button	GIRAF home screen is shown	**Hero**: pass **Tab:** pass **Desire:** pass **Wildfire:** pass
Click an app to open it	The clicked app opens.	**Hero (aSchedule)**: fail* **Tab (bmi):** fail* **Desire (bmi):** fail* **Wildfire:** fail*
Press home	GIRAF home screen is shown	**Hero**: pass **Tab:** pass **Desire:** pass **Wildfire:** pass
Press back.	Nothing happens.	**Hero**: pass **Tab:** pass **Desire:** pass

		Wildfire: pass
Press home.	Nothing happens.	**Hero**: pass **Tab:** pass **Desire:** pass **Wildfire:** pass

Test Results and Observations:

*Open aSchedule or digiPecs (or BMI4Kidz). Enter admin panel. Press home. Open aSchedule. This opens the admin panel. This is unintended behaviour, as children may unitendently enter the admin interface. (reported as issue 43)

Identifier:

TD 06

Program/Module/Object Under Test:

Testing uninstall functionality in girafPlace.

Test Objective:

Test whether applications can be deleted from GirafPlace.

Test Conditions and Intercase Dependencies:

TD01

Test Procedure:

Instruction	**Expected Behaviour**	**Pass / Fail**
Open GIRAF Place, Uninstall an application	Application is uninstalled without warnings or errors.	**Hero:** pass **Wildfire:** pass **Tab:** pass **Desire(bmi):** pass
Open launcher	Launcher does no longer show uninstalled application.	**Hero:** pass **Wildfire:** pass **Tab:** pass **Desire**: pass
Enter admin mode, Open device settings, open package manager	Uninstalled application is not shown in application list.	**Hero**: pass **Wildfire:** pass **Tab:** pass **Desire:** pass
Download uninstalled application from GIRAF Place	Application is downloaded without warnings or errors.	**Hero**: pass **Tab:** pass **Wildfire:** pass **Desire(bmi):** pass

Install application	Application is installed without warnings or errors.	**Hero**: pass **Tab:** pass **Wildfire:** pass **Desire(bmi):** pass
Enter launcher	Application is shown	**Hero**: pass **Tab:** pass **Wildfire:** pass **Desire(bmi):** pass

Test Results and Observations:

The uninstall process works as intended.

Identifier:

TD07

Program/Module/Object Under Test:

GIRAF Place, User Profile

Test Objective:

To check if apps are filtered according to user profile settings.

Test Conditions and Intercase Dependencies:

TD01

Test Procedure:

Instruction	**Expected Behaviour**	**Pass / Fail**
Open home screen.	Home screen is shown.	**Hero**: pass **Wildfire:** pass **Desire:** pass
Go into admin mode.	Admin mode is shown.	**Hero**: pass **Wildfire:** pass **Desire:** pass
Open user profile settings. (through administrative settings)	User profile settings are shown.	**Hero**: pass **Wildfire:** pass **Desire:** pass
Edit capabilities in user profile. Set *canRead* to false.	Ensure all settings are saved when finished.	**Hero**: pass **Wildfire:** pass **Desire:** pass
Press home	Home screen is shown. Ensure no text is shown under applications.	**Hero**: pass **Wildfire:** pass **Desire:** pass

Go into admin mode.	Admin mode is shown.	**Hero**: pass **Wildfire:** pass **Desire:** pass
Start GIRAF Place	Ensure that only applications, that meets the user profile capabilities, or already are installed, are shown.	**Hero**: pass **Desire:** pass **Wildfire:** pass

Test Results and Observations:

Applications were filtered according to specifications.

APPENDIX G

ACRONYMS

IDE Integrated Development Environment

AAC Augmentative and Alternative Communication

PECS Picture Exchange Communication System

OOAD Object Oriented Analysis & Design

ASD Autism Spectrum Disorders

IDA Instant Data Analysis

OS Operating System

SDK Software Development Kit

VDA Video Data Analysis

GUI Graphical User Interface

API Application Programming Interface

OOP Object-Oriented Programming

TTS Text-To-Speech

DBMS Database Management System