AALBORG UNIVERSITY

STUDENT REPORT

GROUP SW601F12

BACHELOR PROJECT

PARROT: Part of the GIRAF system

Authors:
Rasmus D.C. DALHOFF-JENSEN
Jakob JØRGENSEN
Kim Arnold THOMSEN
Christoffer Ilsø VINTHER

Supervisor:
Ulrik NYMAN

June 3, 2012

Student Report

Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Øst
Telephone +45 9940 9940
Telefax +45 9940 9798
http://www.cs.aau.dk

Title: PARROT: Part of the GIRAF system

Subject: Application Development

Semester:
Bachelor Project
SW6, Spring Semester 2012

Project group:
sw601f12

Participants:
Rasmus D.C. Dalhoff-Jensen
Jakob Jørgensen
Kim Arnold Thomsen
Christoffer Ilsø Vinther

Supervisor:
Ulrik Nyman

Number of copies: 6

Number of pages:

Number of appendices: ?

Date: June 3, 2012

Synopsis:

This is a sixth semester report for the AAU Department of Computer Science, and is a part of a multi project with focus on making an $Android^{TM}$ based application for children with Autism Spectrum Disorders. The first part of the report is a common part, which presents the goals of the entire multi project on this semester. Next comes the analysis of our part of the multi project, and following this comes the design for the PARROT application, based on the analysis. Then the implementation of the application is presented, where the functionality is described into details. Finally we will evaluate our project, partially based on a number of tests, and on the system definition.
This report is also written as an aid for the next year's sixth semester students, to help them continue on the project.

PREFACE

This report is written in the sixth semester of the software engineering education at Department of Computer Science, Aalborg University in the Spring of 2012.

The report is written in LaTeX. and it is documentation for the project, made in the period from first of February until fifth of June 2011. The semester topic is "Application Development". The main goal is to gain knowledge about how to make a application. This is done through designing and developing an application for $Android^{TM}$ based tablets.

The reader is expected to have some basic knowledge about application programming and knowledge of how to code in Android.
The following are general unless anything else is mentioned.

- Cites and references to sources are denoted by square brackets containing a number, like this; [18]. Sometimes it is followed by a comment used to specify the reference, like this; [18, Comment]. The letter and number is corresponding to an entry in the bibliography.
- Abbreviations will be presented in extended form the first time they appear.

Throughout the report, the following typographical conventions will be used:

- Omitted unrelated code is shown as '. . . ' in the code examples.

All code examples given in the report are not expected to compile out of context.

Appendices are located at the end of the report.
A installation guide for how to install the PARROT application can be found in the Appendix E.
As this is a multi project report, the Introduction chapter is not written by the PARROT group, but written by all of the participants of GIRAF. As such, the opinions expressed in the Introduction may not match those of PARROT.

This group would like to thank Tove Søby in aiding us in understanding the daily use of the pictogram tool as well as providing an insight into the differences of the children.
Additionally we would like to thank Yuku Sugianto for creating the AmbilWarna library and making it available through to the Apache License 2.0. [11]

The source code for this report is attached on the CD-Rom at the last page of the report. A PDF version of the report is also included at the CD-Rom.

Partisipators of the project:

Kim Arnold Thomsen	Jakob Jørgensen
Christoffer Ilsø Vinther	Rasmus D.C. Dalhoff-Jensen

CONTENTS

WORD DEFINITION

- ASD - Autism Spectrum Disorder.
- Children - Children refers to "Children with ASD".
- Cosmetic - Minor usability error meaning that the test subject is annoyed by the way a function works. Less critical than Severe.
- Critical - Major usability error meaning that the test subject is unable to proceed without getting help. The most critical of usability errors.
- Eclipse - An integrated development environment for Java.
- GIRAF - The name of the project from last year, which we are developing further.
- Guardians - Parents, teachers, caretakers, and educators of children with ASD.
- PARROT - Name of the pictogram application, **P**ictogram **A**ssisting with **R**hetoric **R**easoning **O**r **T**alking
- Pictogram - A graphic symbol that conveys its meaning through its similarity to an object.
- Severe - Usability error meaning that the test subject is confused by the behavior of a function, has trouble understanding how to proceed, or has difficulty using the software. More critical than Cosmetic, less critical than Critical.
- We - In the introduction, it refers to the entire multi project group, as it is written as a common part of all the reports. After the introduction chapter 1, it refers to the individual project group.
- WOMBAT - Name of the timer application, **W**ay **O**f **M**easuring **Ba**sic **T**ime.

CHAPTER 1

INTRODUCTION

In order to describe the context of the system, we – as a multi project group – will in the following state the motivation of the project, the group of people we are aiming at helping, the technological platform chosen, the used development method, followed by a problem definition, a system description and architecture, and the conducted usability test.

1.1 Motivation

As this is a student report written as part of a learning project, we are required to comply with the study regulation. The main areas of focus, according to the study regulation, are: multi-project management and quality assurance in the form of requirements analysis, requirements management, and testing. The goal is to create a comprehensive software system, across multiple project groups, in order to enhance our competences in analysis, design, implementation, and evaluation of software applications in regards to the system requirements[17]. This project builds on top of a previous project, and is further developed, with the aim of having other students continue the development. The goal of the project, we are building on top of, is to create a touch based tablet system to support children and their guardians in everyday scenarios.

1.2 Target Group

Our target group is both children and their guardians. These guardians have certain needs for special tools and gadgets that help to ease the communication between them and the children. Five teachers and educators, who work with children, act as customers. They will provide requirements and information about the institutions' way of working to give us an insight into their daily struggles.

1.2.1 Working with Children with ASD

This section is based upon the statements of a woman with ASD [5], explaining what it is like to live with ASD, and an interview with an educator at Birken, a special kindergarten for children (see appendix A for interview notes).
People with ASD are often more visual in their way of thinking. Rather than visualizing thoughts in language and text, they do it in pictures or visual demonstrations. Pictures and symbols are therefore an essential part of the daily tools used by children and the people interacting with them. Also, children can have difficulties expressing themselves by writing or talking, and can often more easily use electronic devices to either type a sentence or show pictures, to communicate with people around them. Another characteristic of children is their perception of time. Some of them simply do not understand phrases like "in a moment" or "soon", they will need some kind of visual indicator that shows how long time they will have to wait.
Different communication tools for children with autism already exist, but many of them rely on a static database of pictures, and often these has to be printed on paper in order to use them as intended. Other tools, such as hourglasses of different sizes and colors, are also essential when working with children, and these tools are either brought around with the child, or a set is kept every place the child might go, e.g. being at an institution or at home.
There exists tools today which helps the guardians in their daily life, although – as stated in Drazenko's quote – none of them are cost-effective enough to be used throughout the institutions. From the quote, it is clear that there is a need for a more cost-effective solution.

> *The price of the existing solutions are not sufficiently low such that we can afford to buy and use them throughout the institution.*
>
> *- Drazenko Banjak, educator at Egebakken.*

1.3 Target Platform

Since we build upon last year's project, we are bound to use the platform they used, which is tablets running the Android operating system.
In this project we have been provided with five Samsung Galaxy Tab 10.1 devices[10]. The firmware on the tablets is version 3.2. This version, as of project start, is the latest stable version available for these specific tablets. [10]

1.4 Development Method

As a part of the study regulation we have been required to use the same development method in each individual group. Two methods have been considered, *XP* (eXtreme Programming) [18], and Scrum [3].
With the knowledge of both XP and Scrum, we decided in the multi project to use Scrum of Scrums, which is the use of Scrum nested in a larger Scrum project [2].

The reason for choosing Scrum of Scrums is that everyone, at all times, will be able to know what the vision of the project is, and how close every group is to achieving their individual goals of the vision.
Another element of the Scrum method is that a close contact with the customers is maintained. This helps keep the product backlog up to date and correctly prioritized. The customers are presented with the vision of the project, as well as showing the latest release when we have meetings with our customers.
We customized Scrum to fit our project. The changes are as follows:

- The sprint length have been shortened to approximately 7 - 14 half days.
- Some degree of pair programming have been introduced.
- There is no project owner because this is a learning project.
- Everyone is attending the Scrum of Scrums meetings.
- The Scrum of Scrums meetings are only held once at sprint planning.

1.5 Problem Definition

The problem statement is as follows:

> *How can we ease the daily life for children with ASD and their guardians, while complying with the study regulation?*

This problem statement is necessarily vague to allow the individual groups some freedom in their projects, while we maintain the overall structure of the multi project, however there are limiting factors. We are limited by resources and time available, as we are only working on this project for a single semester. However, all work done in this multi project will be passed on to the next line of students, which means we can make a full system design and pass on anything we do not have the time or resources for. This also requires that our work need to be of such quality that it is understandable by students of the same educational level as ourselves.

1.6 System Description

GIRAF is a collection of applications, either fully or partially interdependent, for the Android platform, designed to be used by guardians and children. GIRAF consists of five projects with various degree of interaction. These projects are named Launcher, PARROT, WOMBAT, Oasis, and Savannah. Each of the groups have produced individual products, which are parts of a greater project, GIRAF.
Launcher handles execution of GIRAF apps, and at the same time it provides safety features to ensure that a user that is not authorized to interact with the rest of the system will not be able to do so. When the launcher executes an app, it will provide it with profile information, specifying which child is currently using the app, as well as which guardian is signed in.

PARROT is an app which provides access to pictograms – pictures with associated information such as sound and text – which can be used for communication. PARROT also gives guardians functionality for adding additional pictograms, as well as organizing the pictograms into categories for ease of access, based on the needs of the individual child.
WOMBAT is an app which purpose is to help the children to understand the aspect of time, by visualizing it. WOMBAT provides different ways of displaying time, as well as the possibility to configure the app for the needs of individual children.
Oasis locally stores the data and configuration of the GIRAF platform, and provides an API to access it. The stored data and configurations are synchronized to the Savannah server, if available. In addition, an app is provided for the guardian to access the stored data and configurations.
Savannah provides Oasis with a way to synchronize tablets running GIRAF. Furthermore, a website is provided to ease administration of the synchronized data.

1.7 Architecture

Our System architecture – shown in Figure 1.1 has been designed with simplicity in mind and was greatly inspired by the MVC pattern. This means that the architecture is divided into three layers. The lowest layer is the database where the information is stored. Above this layer is the controller layer which, in the GIRAF platform, is known as Oasis. The controller is responsible for querying the database for information needed in an app and the controller is also responsible for storing information in the database. The last layer is the apps. This division of layers give the GIRAF platform a low cohesion which makes it easier to work with individual parts of the platform independently.
We have chosen to redesign last year's architecture [6] to make it easier to work with. We have simplified the architecture because we feel it is unnecessarily complex.

1.8 Usability Test

As stated in the motivation, quality assurance through testing of the system is required. Therefore a usability test was conducted in order to measure the current usability of the GIRAF platform as a whole, as well as of the individual parts of the platform. Furthermore, the next wave of developers will immediately be able to start correcting the found usability issues.

1.8.1 Approach

The test group consists of the five contact persons. We assess that they, as a test group, are representative. We base this on them being a mix of educators and teachers, with varying computer skills.
They have prior knowledge about the overall idea of the GIRAF platform, and although some of the contact persons had previously informally used some aspects or parts of the system, they had not been exposed to the platform as a whole, and therefore still are of value.
The invitation sent to the test persons can be found in Appendix B.

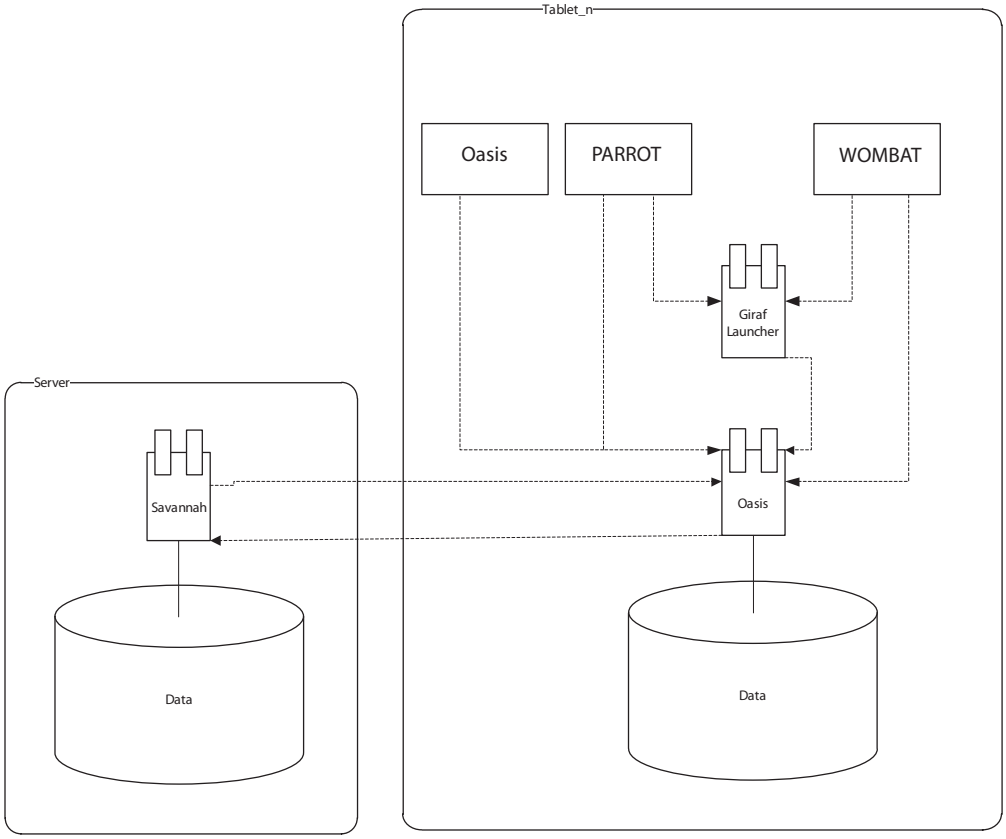

Figure 1.1: The GIRAF architecture

The Instant Data Analysis (IDA) method for usability is chosen. A traditional video analysis method could be used, but since IDA is designed for small test groups, this approach is used. [8]

Setup

The usability test is divided into two tests: A test of three user applications, and a test of two administrative applications. The user applications are: The launcher, PARROT, and WOMBAT. The administrative applications are: The Oasis app and the Savannah web application. Each test is assigned a team to accommodate the need to run two tests simultaneously. The teams are made with respect to the criteria of the Instant Data Analysis process.
Each team consisted of:

- 1 x Test Coordinator
- 1 x Test Monitor
- 1 x Data Logger
- 2 x Observers

The usability lab at Aalborg University is designed with two rooms for usability testing and a control room to observe and record the tests. The two test chambers are assigned a test each and the control room is used to observe both tests as seen in figure 1.2.

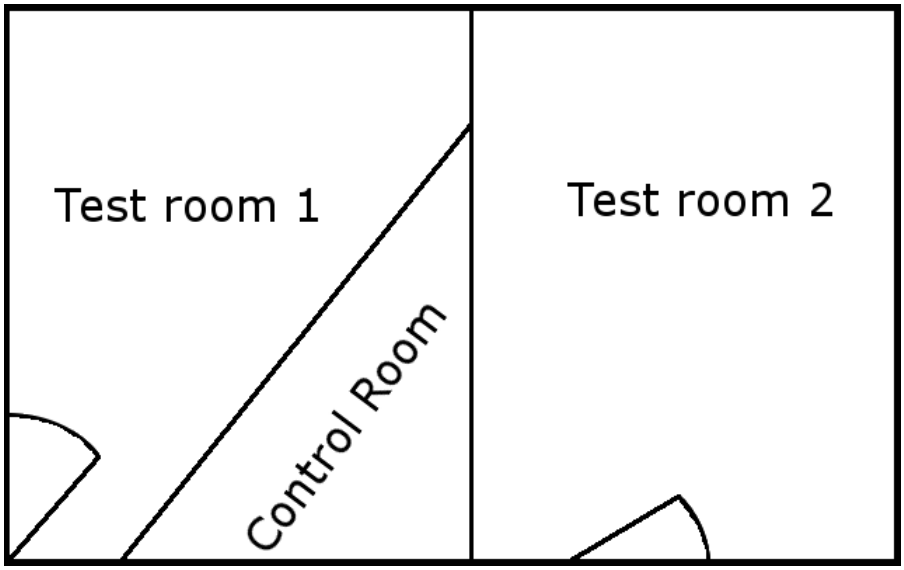

Figure 1.2: An overview of the usability lab at Cassiopeia, Department of Computer Science, Aalborg University.

Execution

Figure 1.3: The schedule of the usability test.

The tests are conducted according to the schedule in Figure 1.3.
Briefing, debriefing, and questionnaire documents can be found in Appendix D, and the results of the test can be found in section 5.5.

CHAPTER 2

ANALYSIS

In this chapter, we will present the results of the initial analysis phase of our project, the daily use of pictograms, as well as the reason for using our product.

To gain greater knowledge on how to work with children, and which tools are in use in the daily work, we have been in contact with Tove Søby, a speech therapy consultant from "Taleinstitutet Aalborg", who works with children with ASD.

Based on the meeting with Tove (see notes in Appendix C), we have learned that one of the tools used in the communication with the children is pictograms. The pictograms are used in a schema for the day, where all their daily activities are listed as pictograms. This gives the children the possibility to go to their schema and see what they are going to do next. The pictograms are also used for direct communication with the children. The pictograms are used by the children to construct sentences to communicate with others, and also to try to teach them to speak if that is a problem for a specific child.

In the use of pictograms, the problem is that they are very space-consuming, and the tools are not very practical to move around, since they consist of a bookcase full of folders with pictograms. Another problem is that when new pictograms are needed, the personnel that needs the new pictogram has to print them out, cut them out in small squares, and laminate them. So in addition to the folders taking up a lot of space, it is also time consuming to produce new pictograms for the children.

We have also learned that there are various degrees of autism, and how it affects the individual child. This is not in focus of our project, but as the children can have various degrees of autism, this means that no two children are the same, as we have been told by our Tove. This means that the tools will need to be able to adapt to the needs of each individual child.

Therefore, based on what we have learned from Tove, and from our own research, it would be practical to have a digital version of the pictograms. So that rather than having a lot of boards and a lot of books filled with pictograms for the children, they would only need a few tablets with the same functionality, just converted to a digital version.

We have now presented the analysis, demonstrating the daily use of pictograms, as well as some of the backsides of the tools currently in use.

CHAPTER 3

APPLICATION DESIGN

In this chapter we will present the design for our Android application and shortly describe which features the application will contain, what the design should look like, and why we have chosen to include these features in our Android application.

Based on last years design for the Digi-Pecs application, modified by the information gathered in the previous chapters, we have come up with the general idea for how we want the design for PARROT to be.(For the Digi-Pecs design [7])
First of all, we want the application to be safe, simple to use, and easy to remember.
By *safe*, we mean that the users of the system should only have access to their own area. For instance, a child should not (usually) have access to the options and administration parts of the application, while guardians should not have access to the profiles of other guardians, or children not under their supervision.

For the system to be *simple to use* and *easy to remember*, we want the application to have a relatively simple user interface. Especially since the application is meant to be used by children. Furthermore, the reason for making this application is to make the daily use of pictograms easier. This goes for the guardians as well. If they can't use the system, the children will not be able to learn how to use it.

Based on the fact that we are making a digital version of the daily use of pictograms, we want to expand the concept of pictogram, meaning that not only will the pictogram contain a picture and a string of text, it will now also contain sound in the form of a pronunciation of the word, as well as a sound effect. Concerning the concepts of usability, we have set the priority of memorability and usability higher than learnability. In other words, the application should be easy to use after the user has been provided with a brief demonstration or a manual, and should be easy to remember afterwards. We haven't specifically designed the application to be self-explanatory but we feel that this would not be a problem since a brief introduction should

be enough to get the user familiar with the how the application works.
It is not our thought that the system has to be intuitive, but it is our opinion that if the user is provided with a short demonstration or manual, they will master the concepts of the application in no time.

Designing the user interface has been an iterative process, as is the method of SCRUM. We started by having a look at the design from the report of the previous year (we were unable to see the actual design) and compared it to the tools shown to us by Tove Søby.
We decided that we wanted our design to look and feel like the real thing, meaning that we want to mimic the physical interaction of moving pictograms on a speech board, so we drew up a quick sketch of how we wanted the system to look.
As seen in Figure 3.1, we wanted a speech board (the part of the application used by the

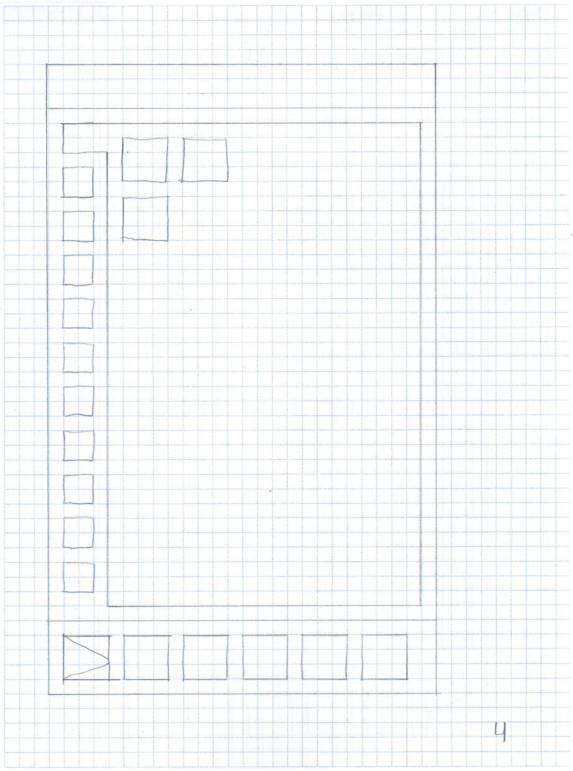

Figure 3.1: Sketch of the speech board tab

children) to show the currently available pictograms, as well as a sentence board to put the pictograms on to create a sentences, and a number of tabs representing different categories of pictograms. We also put a play-button on the design, so that the user could press it to play out the sentence.
The reason for this choice was based on the idea of having the system be as close to the ex-

isting tools as possible. The guardians already have speech boards and sentence boards, and they have lots of folders containing pictograms organized into different categories. This is why we went for the tab style for the categories, as that would make it look more like a folder.
As we began implementing the system, we realized that we had to make several changes. First of all, the size of the initial sentence board was too small, it would have to be bigger in order to make the pictograms easily visible. Also, we had to change the design from tabs to button-like-pictures, as we discovered that Android does not natively support vertical tabs. We also removed the play-button, mainly because our contact said that she would rather have the ability to play each pictogram one at a time, than only the ability to play the entire sentence. Also, the button took up a lot of the already limited space on the interface.

After designing the speech board, we continued on to the parts of the application that are restricted to be used by the guardians. We had to make the design so that the options and the administration where easy to access for the guardian when they need to make changes for the specific child that is using the application.
We tried to see how the application from last year worked, but this was not possible so we began to search for other solutions. We came up with an idea for a tab for the speech board, a tab for options and a tab for the administrations for categories.

The design for the administration for categories can be seen in figure 3.2. First we have a spinner (the Android equivalent of a drop down menu) that contains the available profiles. Below this we have a list of categories for the selected profile. Below that we have a trash bin to dispose of unwanted objects. In the upper right corner we have the selected category information shown, in this box we can change all information regarding the selected category. Below that we have all the pictograms that are found in the selected category.
Now that we had been through the design for the application, we will list the features for the application:

- **Drag and Drop:** This feature allows the user to choose a pictogram, drag it to it's destination, and drop the pictogram. We choose this feature for the application because we want the application to simulate how the pictograms are used normally, meaning that the user has some pictograms and can "pick up" a pictogram and put it on the sentence board.

- **Sound:** This feature enables us to play the audio track that is a part of the pictogram. The feature was chosen because we need a way so that the user of the application can play the sound of the pictogram.

- **Action Bar:** This feature allows us to use tabs in the application. We are using this feature so that the options and administration of categories would be easy to access for the guardian.

- **Color Picker:** This feature gives the user a color scheme so that the user can choose which color the speech board tab should have and the color picker is also used when the guardian chooses color of the categories. We use this feature because one of the

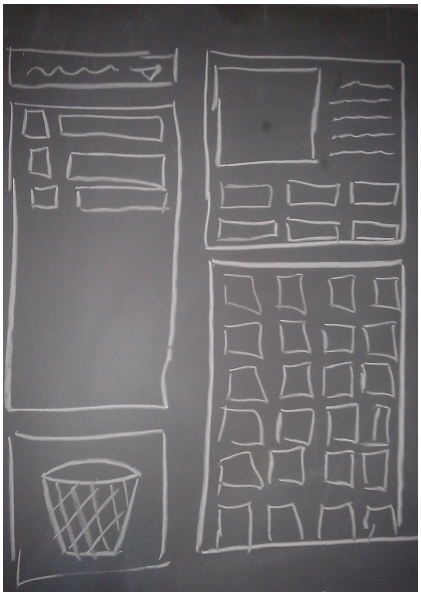

Figure 3.2: Sketch of administration for categories

main points in our application is that we can adjust the application to the needs of each child.

- **Showing Pictograms:** This feature takes the pictograms that are a part of the user's profile, and puts them into the pictogram grid in the speech board tab. This feature is important because it is the part that gives us our pictograms.
- **Sentence Board:** This part of the speech board is where the user can drag pictograms down onto and arranges the chosen pictograms so that the user can make a sentence. It is also where the user can play the audio track in the pictogram. This feature is essential for our application since this is the central feature of this application.
- **Data Management:** This is the part where we access the database to get the profiles and pictograms used by the application.

So with these designs (see figure 3.1 and figure 3.2) we contacted Tove to get some feedback for the designs and she gave us positive feedback, with the comments that the design had to be very flexible and that we should priorities playing a individual pictogram sound over playing an entire sentences, (see notes from the interviews in Appendix C). Based on the interview we adapted our design to fit Tove's ideas, and put it into production.

This implemented design can be seen in Figure 3.3 and Figure 3.4.
The design for our Android application has now been presented with sketches for the different tab of the application. Furthermore, we have presented the features for our application and described how important they are for the end product.

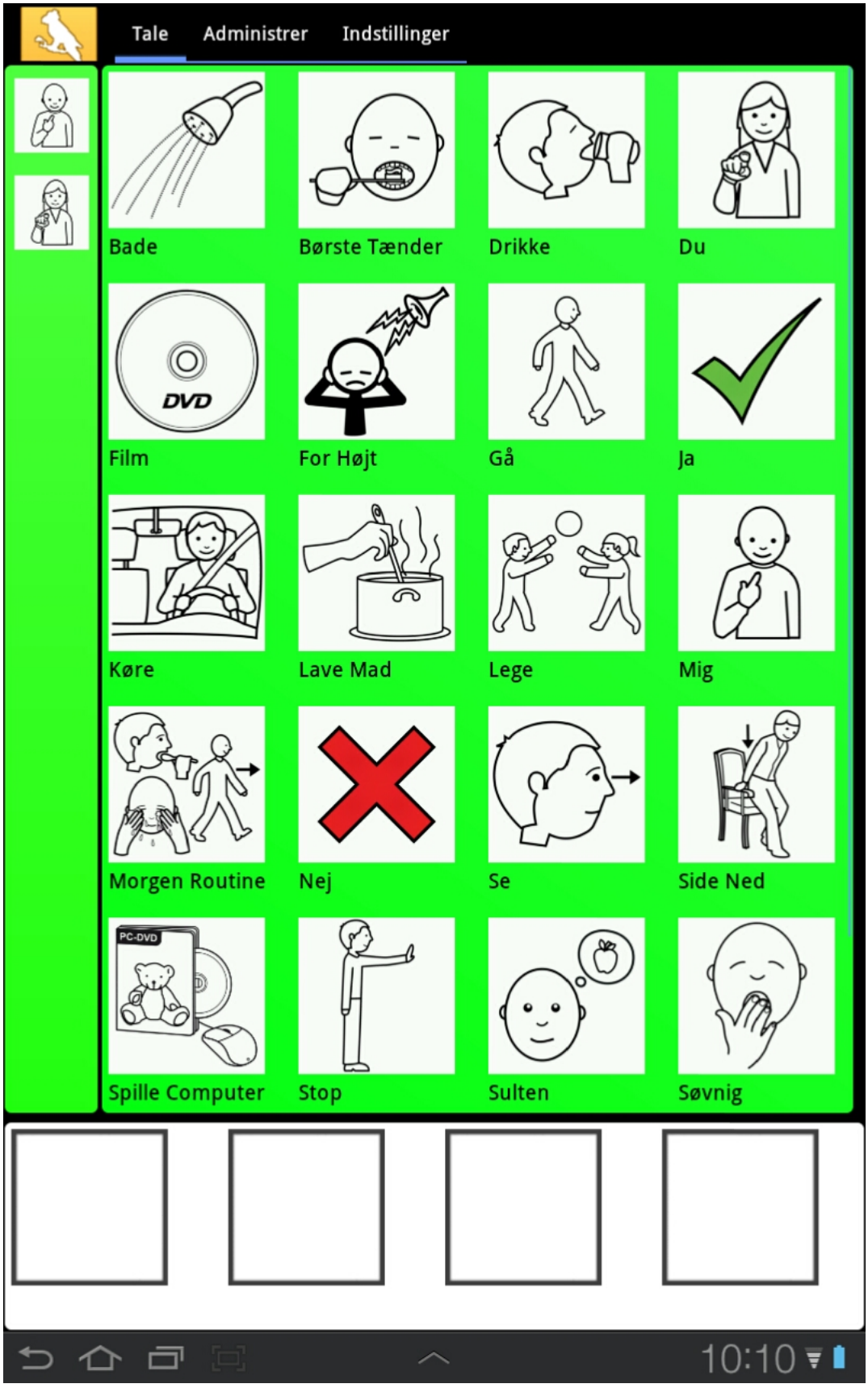

Figure 3.3: The implemented speech board

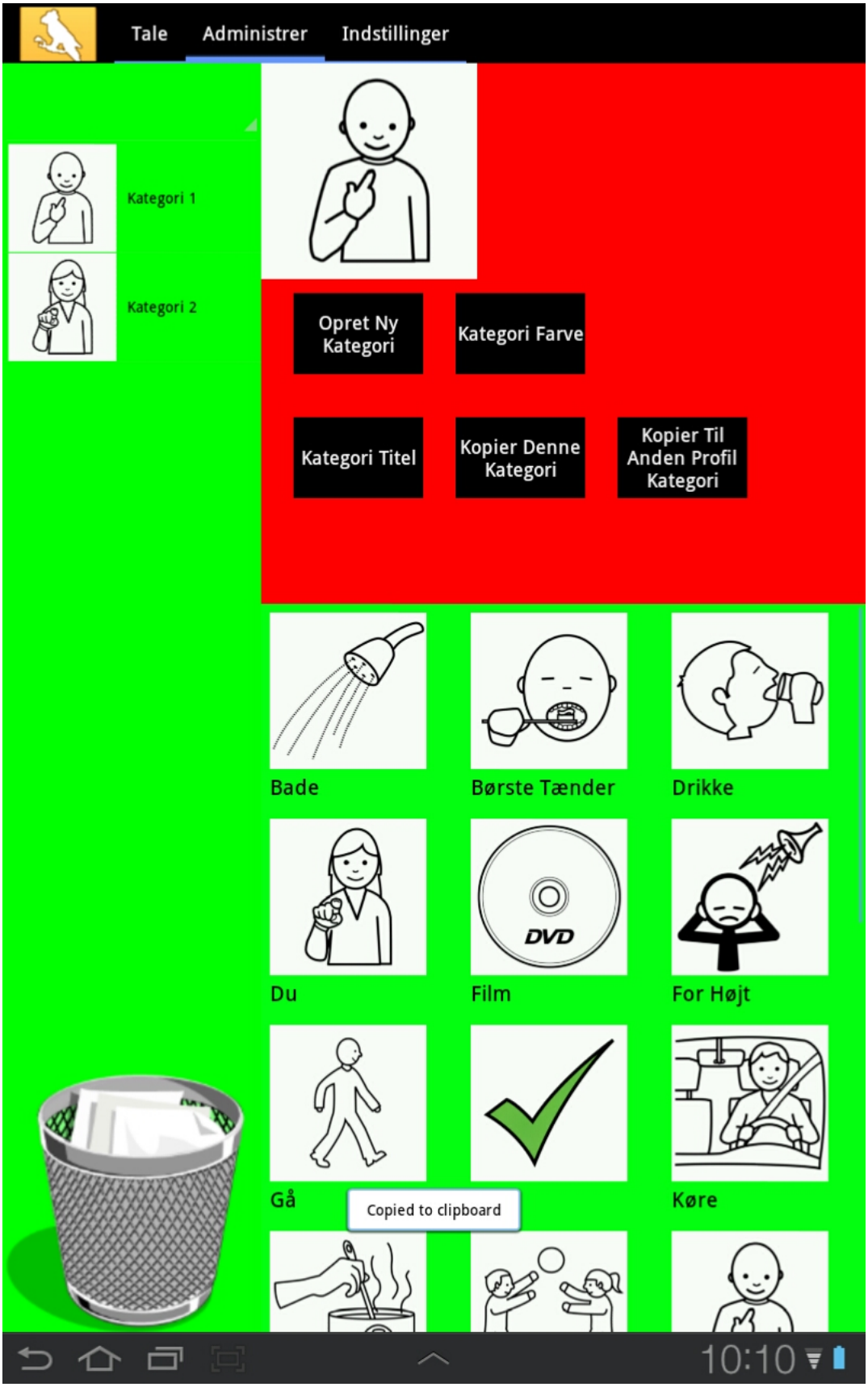

Figure 3.4: The implemented administration tab

CHAPTER 4

IMPLEMENTATION

In the this chapter features of the PARROT will be presented. The source code associated with the software will be presented and described as well. The software is based on the previously analyzed design and calculations. So far it has been described how the model is supposed to work, and now the software implementation will be described in detail.

The features will be described, first as the problem that spurred the feature, and then the solution we thought up. We will then describe how the solution is executed in the PARROT using code segments. Finally we will present notes we might have about the feature, followed by a recommendation of what to read when exploring the problem further.

Figure 4.1 depicts the PARROT project in an UML diagram, showing the class associations .

4.1 XML Layouts

Problem

We need to establish the layout of our application, so that the user has something to interact with.

Solution

The Android SDK allows you to utilize an XML based method of creating the visual layouts and views that are seen in the application. GridView, LinearLayout, and ImageView are all examples of layout elements used in the XML documents. It should also be noted that all attributes and layout parameters of an element are written within the start-tags, “<”. Eclipse with the Android Development Tools installed allows for a drag and drop approach to setting up an XML document, by moving elements on a visual representation of it.

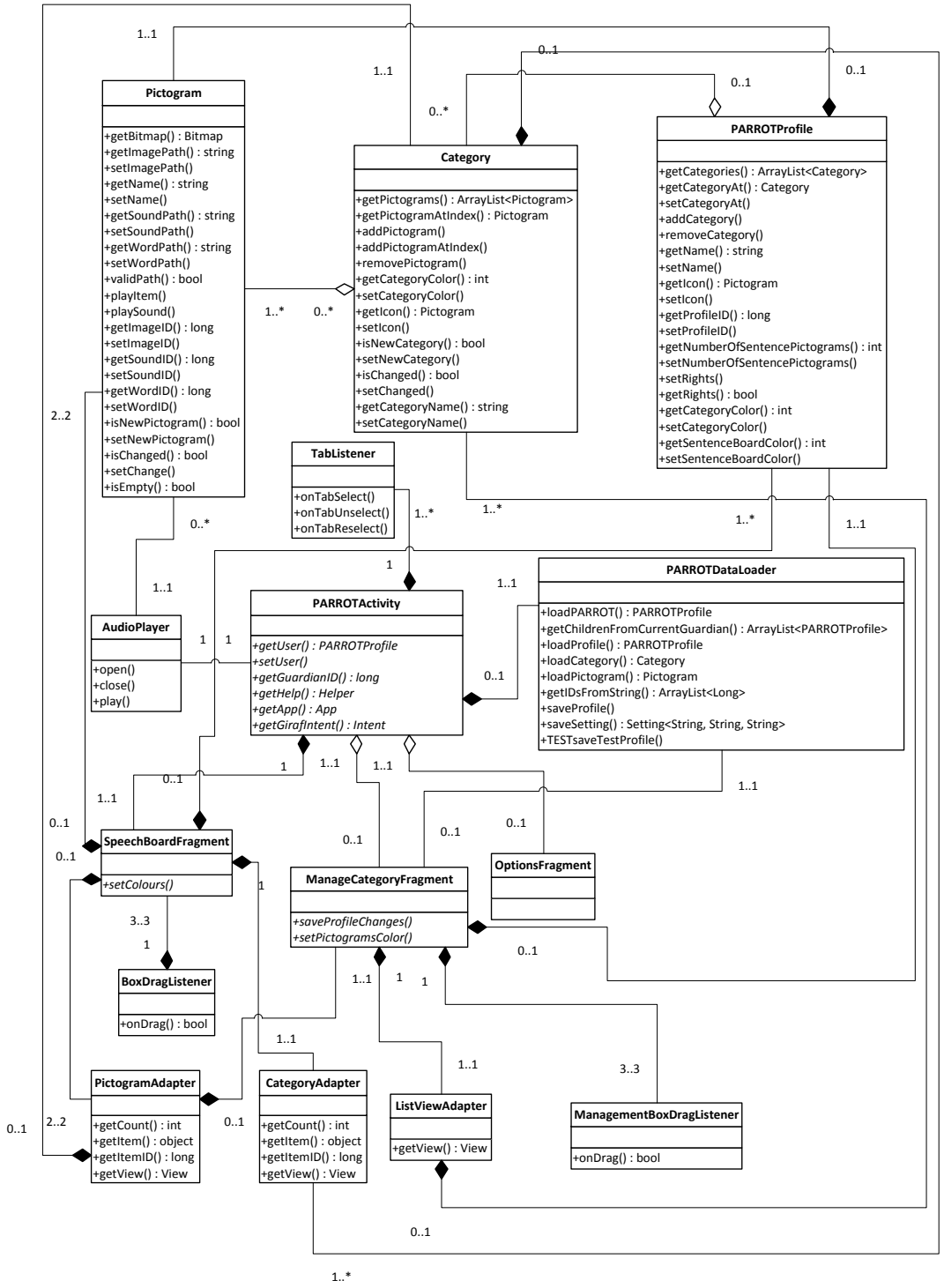

Figure 4.1: UML Diagram of PARROT. Filled diamonds are composition, empty diamonds are aggregation.

Execution

The following is an example of XML code taken from the “categoriesitem.xml” file.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk↵
       /res/android"
3       android:layout_width="match_parent"
4       android:layout_height="125dp"
5       android:orientation="horizontal">
6
7       <ImageView
8           android:id="@+id/catpic"
9           android:layout_width="100dp"
10          android:layout_height="100dp"
11          android:layout_marginLeft="6dp" />
12
13      <TextView
14          android:id="@+id/catname"
15          android:layout_width="wrap_content"
16          android:layout_height="fill_parent"
17          android:layout_marginLeft="10dp"\
18          android:gravity="center_vertical"
19          android:textColor="#FF000000" />
20
21 </LinearLayout>
```

XML Code 4.1: The categoriesitem.xml file

In XML Code 4.1 the root element is a horizontally oriented LinearLayout, as the attributes state. The root element contains two other elements, a TextView and an ImageView, which will be shown in order from left to right because this is a horizontal LinearLayout. Id’s are attached to elements, for reference both within the Java code and the XML document. They are declared by using “@+id/name” and referenced to by using “@id/name”.
In the example, you can see the layout parameters(any) determining the size of the views inside the root, fill/match_parent and wrap_content are two examples of context sensitive sizes, while we also use defined sizes in dp(see Notes. For other definable sizes, see Further Reading.).

Another important layout is the RelativeLayout. This layout does not focus on the order of child elements, rather it allows for attributes such as “android:layout_above” for positioning.

Result

Using XML we have manage to get a functional and viable layout for our application.

Notes

The names of XML documents cannot contain capital letters.
Can be manipulated in activity classes with Java code. See section 4.6 on displaying pictograms, as it contains an example where a view is used by a method.
Unique id's have to be declared the first time they are used in the document, which is not necessarily when they are attached to an Element.
"dp" stands for Density-independent Pixels. Their size depends physical density of the screen.

Further Reading

The guide for various size types can be found on the development website[16], as can the guide on declaring layouts [15].

4.2 Drag And Drop

Problem

We want to be able to drag pictograms from one view onto another. For example we want to drag pictograms from a category onto the sentence board.

Solution

Put simply drag and drop is a 3 step process. Firstly we need to figure which object is dragged, along with whatever information is needed. Secondly we need to show the dragged object. Thirdly we need to handle when the object are dragged and dropped into a view.

We do this by creating listeners to attach to the views we want to drag to or from. These listeners then store information and listen for different actions, including DRAG_STARTED and DROP which helps us define different behavior.

Execution

So far PARROT has 2 tabs which include drag and drop. To optimize the code we have created a listener for each of these tabs. The tabs hold information that the listeners utilize. This information includes the index of the dragged object, and the ID of the view which held the dragged object. The index is used to get all information attached to a given object, while the ID is used to indicate what action is to be performed.
An example on that can be seen in Source Code 4.2, which shows the variables from SpeechBoardFragment.java. These are first initialized to -1 to make sure no mistakes are made.

```
1 public static int draggedPictogramIndex = -1;
2 public static int dragOwnerID =-1;
```

Source Code 4.2: Initializing the variables

In the tabs we also select what views should handle drag and drop. The listeners run for all views of the tab simultaneously. When a object is dropped, all of the handled views will notice the DROP action. Therefore, we have a boolean that handles if the object being dropped is inside a specific view. If the object leaves the view, this is set to false, and if it enters it is set to true, which can be seen in Source Code 4.3. Only if this boolean is set to true do we actually do anything. This secures that an action is only performed once, and in the right view.

```
1 else if (event.getAction() == DragEvent.ACTION_DRAG_ENTERED){
2         insideOfMe = true;
3         }
4 else if (event.getAction() == DragEvent.ACTION_DRAG_EXITED){
5         insideOfMe = false;
6         }
```

Source Code 4.3: The drag entering and exiting the views

When a object is dropped into a view we list a number of possibilities. In each of these possibilities, we check what view we are dropping an object into, and where it is from. What view we drop the object into is handled through the value "self", where the self value is the view that is currently being handled. What view the object is from is gathered from the tab via the former mentioned information. With these two pieces of information, we know where the object is from, and where it has been dropped, and therefore we know what action to perform. These actions vary and some will be described in other sections in chapter 4. After an action has been performed, relevant information is reset.

Result

With this method we are able to drag and drop from and to any view we want. We are also capable of defining what actions are to be performed in each case of drag and drop, which means we can be rather flexible in what functionality we want to add to specific actions. The downside is that the drag and drop classes will be less reusable in others's code. They need a direct connection with the tab they are linked to.

Notes

Much functionality lies in the listeners, which we haven't described here. Mostly because any functionality that is activated by drag and drop are coded into the listeners in some kind of fashion. If one is reading the class file it is recommended to look at some of the other functionality listed in Further Reading.

Further Reading

The listeners are BoxDragListener.java and ManagementBoxDragListener.java. We have used the book Android In Practice [4]. Good guide on how drag and drop functions.
See section 4.7 and section 4.8 for functionality in the listeners.

4.3 Playing Sound

Problem

In our application we are making sentences by using pictograms. These pictograms consists of an image, as well as two optional sounds, as described in chapter 3. The reason for this is that it might help with communication and learning the concepts of what that pictogram illustrates. For this to work, we need an elegant solution that will be easy to use.

Solution

Writing the Pictogram class so that it includes functionality for playing its associated sound files has proven to be a good solution. Not only is it using only a few lines of code, it also makes using the functionality throughout the entire application very easy.

Execution

To use the sound of a pictogram, the application needs to include an audio player so that the sound can be played. Fortunately for the development team, the group who previously worked on the project have already written a functional audio player, which have been easily included in the PARROT application.
In PARROT, the playing of sounds is handled in two different places in cooperation, the Pictogram class (see Source Code 4.4) and the AudioPlayer class (see Source Code 4.5).
The Pictogram class handles the data, and starts up the AudioPlayer wheneven it is needed.

```
1   private void playItem(final String path) {
2      // Running this in the background to keep UI responsive and smooth
3      new Thread(new Runnable() {
4         public void run() {
5            try {
6               AudioPlayer.play(path, null);
7            } catch (Exception e) {
8            }
9         }
10     }).start();
11     //TODO check that the thread is stopped again at some point.
12  }
13
14  public void playSound()
15  {
16     if(soundPath!=null && validPath(soundPath))
17     {
18        playItem(soundPath);
19     }
20  }
21
22  public void playWord()
```

```
23    {
24       if(wordPath!=null &&validPath(wordPath))
25       {
26          playItem(wordPath);
27       }
28    }
```

Source Code 4.4: The play methods from the Pictogram class

For instance, suppose we want to play the word of a pictogram (The word is a sound file with the pronunciation of the pictogram). The first thing we do is call the **playWord** method form the Pictogram in question. This method checks if the pictogram has a word, and if the path of that word corresponds to an existing sound file. If this is correct, it will call the **playItem** with its path as the input.
The **playItem** then starts a new thread in which it makes the AudioPlayer play the corresponding sound. The reason for starting a new thread is that the user interface would become unresponsive while the sound was playing otherwise.
The actual functionality that plays the sound is found in the AudioPlayer class, more specifically in the static **play** method.

```
1    public static void play(String path, final OnCompletionListener listener)
2    {
3       AudioPlayer ap = getInstance();
4
5       try {
6          ap.mMediaPlayer.reset();
7          ap.mMediaPlayer.setDataSource(path);
8          ap.mMediaPlayer.prepare();
9          if (listener != null)
10            ap.mMediaPlayer.setOnCompletionListener(listener);
11         ap.mMediaPlayer.start();
12      } catch (IllegalArgumentException e) {
13         Log.e("sw6", "Media player throw exception, see stack trace");
14         e.printStackTrace();
15      } catch (IllegalStateException e) {
16         Log.e("sw6", "Media player throw exception, see stack trace");
17         e.printStackTrace();
18      } catch (IOException e) {
19         Log.e("sw6", "Media player throw exception, see stack trace");
20         e.printStackTrace();
21      }
22   }
```

Source Code 4.5: The play method from the AudioPlayer class

What happens here is that the internal MediaPlayer object (a native Android class) is reset, fed with data, and prepared for playing the sound. The MediaPlayer object is then started with the **start** method, which causes it to play the sound. If an error occurs, it will be caught, no sound will be played, and the error will be written to the log.

Result

The PARROT application can play sounds for its pictograms as long as they have associated sound files, and will just ignore the request if a pictogram does not have any sound.

Notes

At the current state of the program, the only place in which the functionality for playing sound is being used, is in the speech board tab. Here, the **playWord** method is triggered by clicking on a pictogram on the sentence board.
Currently, the pictograms in the system only have a word-sound associated with them, rather than a sound effect. As such, a new idea will be needed instead of a simple click if both a sound and a word should be able to be played.

Further Reading

For further information about the AudioPlayer class see last year's project [7].

4.4 Tab Navigation

Problem

With major features being either important or big enough to warrant having their own view, we needed to find a way to accommodate this. This is also an important way of avoiding clutter by having too many different features in one view. At the same time we also needed an easy way to isolate features for users who are not supposed to have access to them, as we do not wish for them to be aware of functions they cannot access.

Solution

The action bar with its tab navigation mode, turned out to be the perfect solution. It allowed us to create a tab for each major feature in the parent view. Each major feature has its own fragment, and each fragment is tied to a view. Clicking on the matching tab will prompt the attached fragment to display its view.
The action bar is initialized with the parent activity, as are the tabs. This gives us control over what tabs are going to be present when the application starts, so that a user will never know about the features he is not allowed to use.

Execution

Implementing an action bar first of all means including a TabListener. In this project we did so in a separate class file, using an existing TabListener from (website [13]). Using the TabListener you can instantiate an ActionBar as you can see in Source Code 4.6.

```
1 ActionBar actionBar = getActionBar();
2 actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
3 actionBar.setDisplayShowTitleEnabled(false);
4
5 Tab tab = actionBar.newTab()
6       .setText(R.string.firstTab)
7       .setTabListener(new ↵
            TabListener<SpeechBoardFragment>(this,"speechboard", ↵
            SpeechBoardFragment.class));
8 actionBar.addTab(tab);
```

Source Code 4.6: How to create an action bar and a tab in Android

In order of lines of code:

1. Get the activity's ActionBar through reference.
2. Establish that the action bar is to be used for tabs by setting its Navigation Mode.
3. Determines if an activity should display its title/subtitle.
4. Creating a new tab.
5. Setting the text that should be seen on the tab.
6. Constructing a TabListener so that when you click on this tab or others, the system knows what to do.
7. Finally, add the tab to the ActionBar.

Result

The application can successfully flip between several major features using tabs, as well as hide major features on startup if the user profile does not have permission to use them.

Notes:

Note that in our application, we use if-statements to check if a user has the right to see a tab. The order of the tabs should therefore match up with the order in the Rights Array.
Guides will suggest that you use an override "onCreateView" method to change between tabs with the ActionBar. However, we have concluded that using the override methods "onCreate" and "onResume" was sufficient in our case.

Further Reading:

We used two Android Developer guides which I recommend reading for further development on the application. The first, on ActionBar and its various uses (tab and otherwise), can be found on the homepage [13] as can the second, which covers the fragments that get changed between.[14]

4.5 Color Picker

Problem

Since no two children with autism are the same, our application needs to be customizable. To represent this, we want our application to enable the user to configure the colors of different parts of the application.

Solution

To this problem, the solution is fairly simple. As the WOMBAT group have already found a solution to the problem, we are free to use that solution through data sharing. They have shown us how they use the Android library project called AmbilWarna in their own project, which has been a great help towards implementing it ourselves. The AmbilWarna library provides the application with a dialog box that allows the user to pick a color as an input to the system.

Execution

The use of the AmbilWarna dialog is best seen in the OptionsFragment, more precisely when the **C**hange **C**ategory **C**olour button is clicked.

```
1     ccc.setOnClickListener(new OnClickListener() {
2        public void onClick(View v) {
3           AmbilWarnaDialog dialog = new AmbilWarnaDialog(getActivity(),
4                 PARROTActivity.getUser().getCategoryColor(), new ↵
                      OnAmbilWarnaListener() {
5              public void onCancel(AmbilWarnaDialog dialog) {
6              }
7
8              public void onOk(AmbilWarnaDialog dialog, int color) {
9                 PARROTProfile user = PARROTActivity.getUser();
10                user.setCategoryColor(color);
11                PARROTActivity.setUser(user);
12             }
13          });
14          dialog.show();
15       }
16    });
```

Source Code 4.7: Use of an AmbilWarnaDialog in a button's onClick method.

As seen in the code see Source Code 4.7, if the button is clicked, a new AmbilWarna dialog will pop up. The default selected color of the dialog will be getCategoryColor(see Source Code 4.8).

```
1 PARROTActivity.getUser().getCategoryColor()
```

Source Code 4.8: Getting the color of the category list

This corresponds to the colour that the category list already has for the current user.
If the **Ok** button is pressed on the AmbilWarna dialog, the colour of the category list will be set to the color that the user picked in the dialog.

Result

PARROT gives the user the ability to change the color of different items.

Notes

Currently, the AmbilWarna dialog is used in the OptionsFragment to change the color of some of the items in the SpeechboardFragment. It is also used in the ManagementFragment to change the color of the categories associated with a user.
Note that changing the color of items is only a proof of concept. In the final version of PARROT, the guardians will have the ability to tailor the user interface for the needs of each individual child.

Further Reading

In order to get further insight into the AmbilWarna library we suggest visiting the website [11].

4.6 Displaying Pictograms in a GridView

Problem

We need a way to import pictograms from the tablet storage drive and displaying them in the application so that they are ready to be used. Displaying the pictograms also involves showing the associated text pieces so that one can easily tell what it represents. Areas where we will be displaying pictograms are the speech board and the administration tab.

Solution

Using GridView we can display both the pictograms and its text. Rather than having the GridView display pictograms as ImageViews, we can display a LinearLayout called “PictogramView”. This layout contains an ImageView with a pictogram and a TextView with the associated text.

In that way we are able to show images with text, without having to imprint the text into the image.

In order to prepare the pictograms for display, we have to convert them to Bitmaps before our Pictogram Adapter class can convert them into views.

Execution

In order to determine what to show in a GridView, an adapter is needed. Therefore we constructed the PictogramAdapter in our project, by extending the "BaseAdapter" class. In Source Code 4.9 you can see the getView Method. Its functionality is to return a view for every index in a GridView. These views are the image and text that you see in the application.

```
1   public View getView(int position, View convertView, ViewGroup parent)
2   {
3      ImageView imageView;
4      View view = convertView;
5      TextView textView;
6      Pictogram pct=cat.getPictogramAtIndex(position);
7
8      LayoutInflater layoutInflater = (LayoutInflater) ↵
           context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
9      view = layoutInflater.inflate(R.layout.pictogramview, null);
10
11     imageView = (ImageView) view.findViewById(R.id.pictogrambitmap);
12     imageView.setImageBitmap(pct.getBitmap());
13
14     textView = (TextView) view.findViewById(R.id.pictogramtext);
15     textView.setTextSize(20); //TODO this value should be customizable
16     if(pct.isEmpty() == false)
17     {
18
19        textView.setText(pct.getName());
20     }
21     else
22     {
23        textView.setText("");
24     }
25
26     view.setPadding(8, 8, 8, 8);
27     return view;
28  }
```

Source Code 4.9: The GridView method

We instantiate the different views that we expect to use and start by initializing the Pictogram according to its position in the Category. The Category, in short, is an object containing both information about itself and a list of pictograms. This is followed by creating a "LayoutInflater" which prepares a view for the application. In this case the pictogramview, found through id reference is used to re-initialize the "View" we plan on returning.
The ImageView and TextView are supplied with the data they are to show through method calls. As long as the Pictogram is not an "Empty" Pictogram, "Empty" being our placeholder for pictograms, the View will be given an Pictogram and its name before being returned.

The Source Code 4.10 shows our **getBitmap** method from the pictogram class. This method is important for several reasons. It checks for existing Bitmaps, making sure that the application do not waste time reconverting image files to Bitmaps. It retrieves information from resources and converts it to Bitmaps if the pictogram is supposed to be "Empty", or creates the Bitmap from the image on the tablet storage.

```
1   public Bitmap getBitmap()
2   {
3      if(bitmap == null)
4      {
5         if(isEmpty() == true)
6         {
7            Resources res = parrent.getResources();
8            bitmap = BitmapFactory.decodeResource(res, R.drawable.usynlig);
9         }
10        else
11        {
12           bitmap = BitmapFactory.decodeFile(imagePath);
13        }
14     }
15     return bitmap;
16
17  }
```

Source Code 4.10: The getBitmap method fra the Pictogram class

Result

The result of this is that we are able to use our adapter to show pictograms in GridViews wherever we need them. By storing the bitmap, we can change between views much faster than we could otherwise.

Notes

The input view "convertView" in the getView method is an leftover from a earlier iteration of the method. It no longer serves a purpose but "BaseAdapter" requires that the function exists

with its current inputs.

Further Reading

The PictogramAdapter is based on the adapter shown on the Android developer website [1].

4.7 Sentence Board

Problem:

For children with a hindrance in verbal communication, pictograms are utilized to help them communicate. In the digital version we need some kind of sentence board which has close to the same functionality such pictograms would have if they where used on a physical board. Meaning adding, removing, and reorganize the pictograms in the sentence.

Solution

We generally use drag and drop for most of the functionality. We use it to add, delete, and reorganize the sentence board. We get the pictograms from categories that are shown in a GridView above the sentence board. We can drag pictograms from the gridview into the sentence board to add them. Likewise, if we are to drag a pictogram out from the sentence board, it will be removed from the sentence board. By dragging a pictogram from the sentence board and release the pictogram in the sentence board, it will be shifted to the last position. Dragging a pictogram from the GridView and dropping it on top of a pictogram already in the sentence board, it will simply overwrite the old pictogram. To help this along, we also fill the sentence board with "empty" pictograms, which are there to show that there are currently no pictograms at this specific spaces.

Execution

We create a few long click listeners that utilize our BoxDragListener. These long click listeners are placed for the GridView we collect pictograms from, and the sentence board itself. These listeners are seen in Source Code 4.11.

```
1  pictogramGrid.setOnItemLongClickListener(new OnItemLongClickListener()
2          {
3
4              public boolean onItemLongClick(AdapterView<?> arg0, View view, ↵
                   int position, long id)
5              {
6                  draggedPictogramIndex = position; //TODO make sure that ↵
                       position is the index of the pictogram
7                  dragOwnerID = R.id.pictogramgrid;
8                  ClipData data = ClipData.newPlainText("label", "text"); ↵
                       //TODO Dummy. Pictogram information can be placed here ↵
                       instead.
```

```
9                DragShadowBuilder shadowBuilder = new DragShadowBuilder(view);
10               view.startDrag(data, shadowBuilder, view, 0);
11               return true;
12           }
```

Source Code 4.11: Long clicking a pictogram

Here we also draw the shadow, so that the user can see that they are dragging a pictogram. The rest of the functionality is in the BoxDragListener. See section 4.2.
When we know that a pictogram is from outside the sentence board and is dropped into the sentence board, we know that we are to add the pictogram to the sentence. At the moment this only occurs when the pictogram are dragged from the GridView. If the pictogram is dropped on top of an already existing pictogram in the sentence board, it will overwrite the pictogram on the sentence board.
If it is dropped anywhere else in the sentence board, it will add itself to the last entry of the sentence. In this situation, we need to know where the pictogram is dropped. We do this by getting the position of the drop, and then what index the object at this position in the sentence board has. This can be seen in Source Code 4.12.

```
1  if( self.getId() == R.id.sentenceboard && SpeechBoardFragment.dragOwnerID ↵
       != R.id.sentenceboard) //We are about to drop a view into the ↵
       sentenceboard
2              {
3                 GridView speech = (GridView) ↵
                      parrent.findViewById(R.id.sentenceboard);
4                 int x = (int)event.getX();
5                 int y = (int)event.getY();
6                 int index = speech.pointToPosition(x, y);
```

Source Code 4.12: Dropping a view into the sentence board

We then check if there already is a pictogram in the given place. We can do this by checking if the current pictogram is the “empty” pictogram. If it is not, we know that there already is a pictogram, and we replace it. This is seen in Source Code 4.13.

```
1  if(SpeechBoardFragment.speechBoardCategory.getPictogramAtIndex
2     (index).isEmpty() == false) //Replaces a pictogram already in the ↵
          sentencebord
3                  {
4                     //Removes the pictogram at the specific index
5                     SpeechBoardFragment.speechBoardCategory.removePictogram(index);
6
7                     SpeechBoardFragment.speechBoardCategory.addPictogramAtIndex
8                        (draggedPictogram, index); //add the pictogram at ↵
                             the specific position
```

```
9                   }
```

Source Code 4.13: Replacing a pictogram in the sentence board

If it is the "empty" pictogram, it will try to place the pictogram furthest to the left of the sentence. We do this with a while loop that runs for the duration of spaces in the sentence board. We start at the leftmost entry in the sentence board if its empty, and continue to the right. When an empty place is found, the pictogram is placed here and the loop breaks. This is shown in Source Code 4.14.

```
1  else
2     {
3     int count = 0;
4     while (count < numberOfSentencePictograms)
5        {
6
7        if ↵
             (SpeechBoardFragment.speechBoardCategory.getPictogramAtIndex(count).isEmpty() ↵
             == true)
8           {
9           SpeechBoardFragment.speechBoardCategory.removePictogram(count); ↵
                //Removes the pictogram at the specific index
10          SpeechBoardFragment.speechBoardCategory.addPictogramAtIndex(draggedPictogram, ↵
                count); //add the pictogram at the specific position
11          break;
12          }
13          count++;
14       }
15    }
```

Source Code 4.14: Finding an empty space

When we know that a pictogram is from the sentence board and is dropped into the sentence board, it is because we are about to rearrange the pictograms in the sentence board. We use the same method as we did when we where about to drag something from the GridView into the sentence board.
When we know that a pictogram is from the sentence board and dropped somewhere else, we simply remove it from the sentence board. This is already done in the DRAG_STARTED action. Whenever a pictogram is being dragged from the sentence board, it is removed from it. It can then be placed again via the DROP action. If this happens outside the sentence board though, it is simply removed. This method removes the pictogram, and shifts all pictograms that are to the right of it, one to the left, so that no spaces are left in the sentence. The deletion of a pictogram is seen in Source Code 4.15.

```
1  if (event.getAction() == DragEvent.ACTION_DRAG_STARTED){
```

```
2           if(self.getId() == R.id.sentenceboard && ↵
                SpeechBoardFragment.dragOwnerID == R.id.sentenceboard)
3           {
4              draggedPictogram = ↵
                   SpeechBoardFragment.speechBoardCategory.getPictogramAtIndex
5              (SpeechBoardFragment.draggedPictogramIndex);
6
7              if(draggedPictogram.isEmpty()==true)
8              {
9                 //Do not allow dragging empty pictograms
10             }
11             else
12             {
13                GridView speech = (GridView) ↵
                      parrent.findViewById(R.id.sentenceboard);
14
15                SpeechBoardFragment.speechBoardCategory.removePictogram
16                (SpeechBoardFragment.draggedPictogramIndex);
17
18                SpeechBoardFragment.speechBoardCategory.addPictogram(new ↵
                      Pictogram("#usynlig#", null, null, null, parrent));
19
20                speech.setAdapter(new ↵
                      PictogramAdapter(SpeechBoardFragment.speechBoardCategory, ↵
                      parrent));
21             }
22          }
```

Source Code 4.15: Adding empty pictograms to the sentence board when pictograms are removed.

Result

We can now add and remove pictograms from the sentence board. We can replace pictograms already on the board. The pictograms will always be shifted so that there are no spaces between picograms. It is possible to rearrange the pictograms.

Notes

It should be noted that we have chosen that the pictograms shift to the left whenever they can. This is a design choice that can be changed if further study shows it to be favorable to do otherwise. In the same way we have made the choice that pictograms are to be overwritten. Another alternative would be to shift all pictograms to make room for the new pictogram. This is also something that can be changed if further study shows that it is necessary.

Further Reading

These functionalities are closely bound to the Drag And Drop functionality see section 4.2.

4.8 Managing Categories

Problem

When more and more pictograms are added to PARROT, categories can help manage and organize these pictograms. Categories help manage the different users's needs, so for every user, the categories can differ. We therefore need to be able to configure categories to suit the different users's needs, as well as createing new categories. There is also the need to remove unwanted categories from a user's profile.

Solution

To do this, we use the combination of buttons, and drag and drop. Functionality like deleting categories and pictograms, and adding pictograms from one category to another is done via drag and drop. Creating new categories and changing information about the categories is done using buttons.

Execution

We will first describe the drag and drop functionalities. When we want to delete a specific pictogram from a category, we choose the pictogram from the gridview, and drag it to the trashbin. In the DROP part of the listener, we check if the drop zone is the trashbin, and that the dragged object is from the GridView. When we know this, we create a temporary category, where we save the information of the category we are currently modifying. We then remove the pictogram, and overwrite the old category with this new information. This can be seen in Source Code 4.16.

```
1  if(self.getId()==R.id.trash && ManageCategoryFragment.catDragOwnerID == ↵
       R.id.pictograms) //We are to delete a pictogram from a category
2              {
3                 Category temp = ↵
                      ManageCategoryFragment.profileBeingModified.getCategoryAt
4                    (ManageCategoryFragment.currentCategoryId);
5
6                 temp.removePictogram(ManageCategoryFragment.draggedItemIndex);
7
8                 ManageCategoryFragment.profileBeingModified.setCategoryAt
9                    (ManageCategoryFragment.currentCategoryId, temp);
10
11                pictograms.setAdapter(new ↵
                      PictogramAdapter(ManageCategoryFragment.profileBeingModified.
12                   getCategoryAt(ManageCategoryFragment.currentCategoryId), ↵
                         parrent));
```

```
13            }
```

Source Code 4.16: Deleting a pictogram from a category

When we want to copy a picogram from one category into another category, we choose the category we want to copy from, and drag the chosen pictogram onto the category that we want to copy the pictogram to. Still in the DROP part of the listener, we check if the drop zone is the categories, and if the dragged object is from the GridView showing the pictograms. To know what category we are dropping the pictogram into, we need to know the index of the category, which we do by coordinates of the drop, and referencing to the index of the list. We then add the pictogram to the category at the index. This can be seen in Source Code 4.17.

```
1  else if(self.getId()==R.id.categories && ↵
       ManageCategoryFragment.catDragOwnerID == R.id.pictograms) //We are to ↵
       copy a pictogram into another category
2             {
3
4                draggedPictogram = ↵
                     ManageCategoryFragment.profileBeingModified.getCategoryAt
5                    (ManageCategoryFragment.currentCategoryId).getPictogramAtIndex
6                    (ManageCategoryFragment.draggedItemIndex);
7
8                ListView categories = (ListView) ↵
                     parrent.findViewById(R.id.categories);
9                int x = (int)event.getX();
10               int y = (int)event.getY();
11               int index = categories.pointToPosition(x, y);
12
13               Category temp = ↵
                     ManageCategoryFragment.profileBeingModified.getCategoryAt(index);
14
15               temp.addPictogram(draggedPictogram);
16
17               ManageCategoryFragment.profileBeingModified.setCategoryAt(index, ↵
                     temp);
18
19            }
```

Source Code 4.17: Copying a pictogram into another category

When we want to copy all pictograms from one category into another category, we first choose the category which we want to copy to. Then we drag the category into the GridView showing the pictograms. Also in the DROP part of the listener, we check if the drop zone is the GridView showing the pictograms, and that the dragged object is from the categories. We then initialize two category objects. One containing the category from which we want to copy, which we

call “categoryCopiedFrom”, and one containing the category we want to copy to ,which we call temp. We then go through the pictograms in “categoryCopiedFrom”, and add these pictograms to temp. When all pictograms are copied, we overwrite the old category we wanted to copy from with the new temp category. This can be seen in Source Code 4.18.

```
1  else if(self.getId()==R.id.pictograms && ↵
       ManageCategoryFragment.catDragOwnerID == R.id.categories) //We are to ↵
       copy a category into another category
2          {
3
4             Category categoryCopiedFrom = ↵
                  ManageCategoryFragment.profileBeingModified.getCategoryAt
5             (ManageCategoryFragment.draggedItemIndex);
6
7             Category temp = ↵
                  ManageCategoryFragment.profileBeingModified.getCategoryAt
8             (ManageCategoryFragment.currentCategoryId);
9
10            for(int i = 0; i < ↵
                  categoryCopiedFrom.getPictograms().size(); i++)
11            {
12               temp.addPictogram(categoryCopiedFrom.getPictogramAtIndex(i));
13            }
14
15            ManageCategoryFragment.profileBeingModified.setCategoryAt
16               (ManageCategoryFragment.currentCategoryId, temp);
17
18            pictograms.setAdapter(new PictogramAdapter
19               (ManageCategoryFragment.profileBeingModified.getCategoryAt
20               (ManageCategoryFragment.currentCategoryId), parrent));
21         }
```

Source Code 4.18: Copying a category into another category

When we want to delete a category, we drag the chosen category into the trashbin. In the DROP part of the listener, we check if the drop zone is the trashbin, and that the dragged object is from the category grid. We then remove the category from the profile that we are currently modifying. This can be seen in Source Code 4.19:

```
1  else if(self.getId()==R.id.trash && ManageCategoryFragment.catDragOwnerID ↵
       == R.id.categories) //We are to delete a category
2          {
3             ManageCategoryFragment.profileBeingModified.removeCategory
4             (ManageCategoryFragment.draggedItemIndex);
5
6             categories.setAdapter(new ListViewAdapter(parrent, ↵
                  R.layout.categoriesitem,
```

```
7                   ManageCategoryFragment.profileBeingModified.getCategories()));
8           }
```

Source Code 4.19: Deleting a category

When we want to change the icon of a category, we choose a pictogram from the GridView and drag into the space for the icon, which replaces the current icon. In the DROP part of the listener, we check if the drop zone is the Icon ImageView, and the dragged object is from the GridView showing the pictograms. If so, we notice what index the dragged object has, and create a temporary category. In this category we set its icon to be that of the dragged object. We then overwrites the old category with the temporary category. Then we begin to draw the new icon in the ImageView so that it can be seen. This can be seen in Source Code 4.20.

```
1  else if(self.getId()==R.id.categorypic && ↵
       ManageCategoryFragment.catDragOwnerID == R.id.pictograms) //We are to ↵
       change the icon of the category
2           {
3              draggedPictogram = ↵
                   ManageCategoryFragment.profileBeingModified.getCategoryAt
4              (ManageCategoryFragment.currentCategoryId).getPictogramAtIndex
5              (ManageCategoryFragment.draggedItemIndex);
6
7              Category tempCat = ↵
                   ManageCategoryFragment.profileBeingModified.getCategoryAt
8              (ManageCategoryFragment.currentCategoryId);
9
10             tempCat.setIcon(draggedPictogram);
11
12             ManageCategoryFragment.profileBeingModified.setCategoryAt
13             (ManageCategoryFragment.currentCategoryId, tempCat);
14
15             ImageView icon = (ImageView) ↵
                   parrent.findViewById(R.id.categorypic);
16
17             icon.setImageBitmap
18             (ManageCategoryFragment.profileBeingModified.getCategoryAt
19             (ManageCategoryFragment.currentCategoryId).getIcon().getBitmap());
20
21             ListView list = (ListView) ↵
                   parrent.findViewById(R.id.categories);
22
23             list.setAdapter(new ListViewAdapter(parrent, ↵
                   R.layout.categoriesitem,
24             ManageCategoryFragment.profileBeingModified.getCategories()));
25          }
```

Source Code 4.20: Changing the category icon.

We will now describe the functions that are activated via buttons. When we want to create a new category, we simply push the "create new category" button. We use a onClikListener in ManageCategoryFragment. When the button is pushed it creates a new category for the profile currently being chosen, as well as some information needed to create such a category. We create a empty pictogram to be used as its icon, sets its name to "kategori navn", and its color to red. This can be seen in Source Code 4.21.

```
1  createNewCategory.setOnClickListener(new OnClickListener()
2        {
3           public void onClick(View v)
4           {
5              Pictogram pictogram = new Pictogram("#usynlig#", null, null, ↵
                   null, parrent);
6
7              Category cat = new Category("Kategori Navn", 0xffff0000, ↵
                   pictogram);
8
9              profileBeingModified.addCategory(cat);
10
11             ListView categories = (ListView) ↵
                   parrent.findViewById(R.id.categories); //Redrawing the ↵
                   categories
12
13             categories.setAdapter(new ListViewAdapter(parrent, ↵
                   R.layout.categoriesitem, ↵
                   profileBeingModified.getCategories()));
14             //Adapter for the category GridView
15          }
16       });
```

Source Code 4.21: Creating a new category

When we want to change the color of a category we push the "Change Category Color" button. For this we also use a onClikListener, which activates a pop up screen where it is possible to choose the color. This pop up screen is described in section 4.5. We then save the output from this into the category as its background color.

Result

We are now able to manipulate, organize and manage categories. We can create and delete categories, add and delete pictograms to them, add all pictograms from one category into another, and change the categories icon as well as background color.

Notes

The functionality of changing a category's icon was first designed to be a button. We changed this while coding because it seemed more intuitive.
Unfortunately, there is some functionality that is not yet finished. And although the buttons are in the design they do not do anything. These functionalities include the spinner, which should have made it possible to change what profile was being modified, as well as the buttons making it possible to change the title of categories, copy a category from another profile, and copy the current category to another profile.

Further Reading

Drag and drop are used in these functions. It can be found at section 4.2.

4.9 Data management

Problem

In PARROT, we uses a lot of classes specifically designed for any given situation, such as the Pictogram class, the Category class, and the PARROTProfile class. These classes contain the exact amount of information needed in the application, and have been tailored to make it easier for the developpers to understand the flow of the program.
However, since all the data used in PARROT is provided by the local database through the Admin functionality, there are some things to take into account. The primary problem is that the data classes provided by the admin does not match the ones used in PARROT, and therefore needs to be transformed into PARROT objects before they can be used.

Solution

In PARROT, we have solved the problem of data transformation by writing a class called PARROTDataLoader. The purpose of this class is to handle all interaction between the PARROT application and the Admin interface. An object of this class is used whenever it is necessary to transfer information between PARROT and the database. For instance, whenever the application is started, all information about the current user is loaded from the database.

Execution

To get an idea of how the system works, we will show the code for how categories and pictograms are handled.
The first piece of code to be shown is **loadProfile** see Source Code 4.22.

```
1  public PARROTProfile loadProfile(Long childId,Long appId)
2     {
3        Profile prof;
4
5        if(childId !=null && appId !=null)
```

```
6       {
7          prof = help.profilesHelper.getProfileById(childId); //It used to ↵
               be "currentProfileId"
8
9          Pictogram pic = new Pictogram(prof.getFirstname(), ↵
               prof.getPicture(), null, null); //TODO discuss whether this ↵
               image might be changed
10         PARROTProfile parrotUser = new PARROTProfile(prof.getFirstname(), ↵
               pic);
11         parrotUser.setProfileID(prof.getId());
12         Setting<String, String, String> specialSettings = ↵
               app.getSettings();//This object might be null
13         if(specialSettings != null)
14         {
15            //Load the settings
16            parrotUser = loadSettings(parrotUser, specialSettings);
17
18            //Add all of the categories to the profile
19            int number = 0;
20            String categoryString=null;
21            while (true)
22            {
23               //Here we read the pictograms of the categories
24               //The settings reader uses this format :
25               // category +number | cat_property | value
26               try
27               {
28                  categoryString = ↵
                        specialSettings.get("category"+number).get("pictograms");
29               }
30               catch (NullPointerException e)
31               {
32                  //the value does not exist, so we will not load anymore ↵
                        categories
33                  break;
34               }
35
36               String colourString = ↵
                     specialSettings.get("category"+number).get("colour");
37               int col=Integer.valueOf(colourString);
38               String iconString = ↵
                     specialSettings.get("category"+number).get("icon");
39               String catName = ↵
                     specialSettings.get("category"+number).get("name");
40               parrotUser.addCategory(loadCategory(catName,categoryString,
41                  col,iconString));
42               number++;
43            }
44
```

```
45              return parrotUser;
46          }
47          else
48          {
49              //If no profile is found, return null.
50              //It means that the launcher has not provided a profile, either ↵
                    due to an error, or because PARROT has been launched ↵
                    outside of GIRAF.
51              return null;
52          }
53      }
54      //If an error has happened, return null
55      return null;
56
57
58  }
```

Source Code 4.22: The loadProfile method from the PARROTDataLoader

First, we check if the ID's have a value different from null. If they do not, an error has happened, and we abort the operation. If they do, we continue and load a profile from the database corresponding to the child currently using PARROT.
Then, we begin the the conversion. First a Pictogram object is made, which is fed with the first name of the child, as well as the path to the picture of the child. This pictogram will serve as the child's identity icon. Then, a PARROTProfile is instantiated with the name of the child, as well as its Pictogram icon.
In order to go on, we load a Setting object from the database. We will not go in depth with this object, as that is the topic of another report (the OASIS report – not yet published), but suffice to say that it is a wrapper class for a hash table which we use to store data.
The Setting object "specialSettings" returned from admin contains information unique to the current child user of the PARROT application, such as visual settings, and pictogram categories. Now, supposing "specialSettings" is not null (in which case we would abort and return null), we load the user specific settings with **loadSettings**, and goes on to load the categories.
The way we load categories is special, so it will be described in detail. First of, categories are stored in the specialSettings hash-table in the format:

$$category\_number|category\_property|value$$

So what we do is iterate our way through the "specialSettings" object. We start by trying to get the pictograms corresponding to Category number 0, which will be the first Category in the list. If the Category exists, we will get a string containing information about the pictograms. If not, we have already found all the categories (in this case none) belonging to the current user. After that, we use the same procedure to get the Category's color, icon and category name. With this information now at hand, we call the **loadCategory** method, and adds the resulting Category to the PARROTProfile "parrotUser", which is the object representing the current user.

After this, we will increment the number, so that we will look for Category number 1. We will continue to increment the number until we have found all the categories, and after that, we will return the parrotUser object.

Following the description of the loadProfile method, we will describe **loadCategory** see Source Code 4.23.

```
1   public Category loadCategory(String catName, String pictureIDs,int ↵
        colour,String iconString)
2   {
3      Long iconId = Long.valueOf(iconString);
4      Category cat = new Category(catName, colour, loadPictogram(iconId));
5      ArrayList<Long> listIDs = getIDsFromString(pictureIDs);
6      for(int i = 0; i<listIDs.size();i++)
7      {
8         cat.addPictogram(loadPictogram(listIDs.get(i)));
9      }
10     return cat;
11  }
```

Source Code 4.23: The loadCategory method from the PARROTDataLoader

The purpose of **loadCategory** is to instantiate a Category object, fill it with pictograms, and return it to the system. The method starts by converting the "iconString" into a Long, which is the ID of a Media object in the database.
After that, we construct a new Category object "cat" from the name and color given by inputs of **loadCategory**, as well as the icon of the Category. Then comes the part where we transform the string "pictureIDs" into a list of numbers. This is done by a method called **getIDsFromString**. The functionality of this method is described with an example: Suppose a Category is going to contain the items 5, 23, 12, and 18. Then the string pictureIDs will look like this:

5#23#12#18$

The method **getIDsFromString** will then return the list
$\{5, 23, 12, 18\}$
which corresponds to the IDs.
When we have all the IDs, the method will go through the list, and return the pictogram corresponding to the given ID. Finally, the Category object "cat" is returned.

After having described how we load a Category into the system, we will described how the **loadPictogram** method works, see Source Code 4.24.

```
1   public Pictogram loadPictogram(long id)
2   {
3      Pictogram pic = null;
```

```
4       Media media=help.mediaHelper.getSingleMediaById(id); //This is the ↵
            image media //TODO check type
5
6       List<Media> subMedias = help.mediaHelper.getSubMediaByMedia(media); ↵
            //TODO find out if this is ok, or if it needs to be an ArrayList
7       Media investigatedMedia;
8       String soundPath = null;
9       String wordPath = null;
10      long soundID = -1; //If this value is still -1 when we save a media, ↵
            it is because the pictogram has no sound.
11      long wordID = -1;
12
13      if(subMedias != null) //Media files can have a link to a sub-media ↵
            file, check if this one does.
14      {
15         for(int i = 0;i<subMedias.size();i++)
16         {
17            investigatedMedia =subMedias.get(i);
18            if(investigatedMedia.getMType().equals("SOUND"))
19            {
20               soundPath = investigatedMedia.getMPath();
21               soundID= investigatedMedia.getId();
22            }
23            else if(investigatedMedia.getMType().equals("WORD"))
24            {
25               wordPath = investigatedMedia.getMPath();
26               wordID = investigatedMedia.getId();
27            }
28         }
29      }
30      pic = new Pictogram(media.getName(), media.getMPath(), soundPath, ↵
            wordPath);
31      //set the different ID's
32      pic.setImageID(id);
33      pic.setSoundID(soundID);
34      pic.setWordID(wordID);
35
36      return pic;
37   }
```

Source Code 4.24: The loadPictogram method from the PARROTDataLoader

The responsibility of the method loadPictogram is to return a pictogram, given its input. The input is the ID of a Media object of the type IMAGE.
We start by instantiating an empty Pictogram object "pic".
Then, we use the admin funtionality given by OasisLib to get the Media object "media", which is the image part of the pictogram. Then, we use the newly loaded object media to get a list

of Medias called “subMedias”. This list contains the sound and word parts of the pictogram. Note that this list can be empty.
Now we create 5 new objects, a Media called “investigatedMedia”, two String objects called “soundPath” and “wordPath”, and Long values “soundID” and “wordID”, which are both set to -1. If the list “subMedias” is not empty, we will iterate our way through it.
We set the value of “investigatedMedia” to that of the Media object at the current position in “investigatedMedia”, and prepare to determine what kind of object it is. We look at “investigatedMedia”, and determine if its type is SOUND, or if it is WORD. If the type is SOUND, we set “soundPath” to the path of the media, and set “soundID” to the ID of the media. If the type is WORD, we do the same but for “wordPath” and “wordID” instead.
Note that it is possible that the type of “investigatedMedia” is a completely different type. In this case, the association is made by one of the other applications, and we simply ignore the media and continue through the list.
When we have gone through the list, we initialize “pic” as a new Pictogram from the name of media, the path of “media”, “wordPath”, and “soundPath”. Finally, we set the IDs of “pic” to those of the different medias and return “pic”.
Note that if an ID is -1 at this point, or a path is null, it simply means that the pictogram in question does not contain either a word or a sound.

Result

The PARROTDataLoader class provides PARROT with objects that can interact with the local database, either to read from the database, or write to it.

Notes

In the current version of PARROT, the connection to GIRAF launcher is not yet fully made, so instead of loading the Category chosen by the launcher, PARROT instead starts by creating a profile, saving it to the database, and then loading it again. The functions in the PARROTDataLoader should be able to handle these changes, but will need to be improved to handle more chances of null pointers.
If changes are made to the database, PARROTDataLoader needs to be changed accordingly, so that it will continue to work.
Finally, PARROT does not contain any save functionality in its user interface. This should be added at a later date. Also, adding a **saveProfile** call to PARROT’s **onPause** method is a way of improving data security.

Further Reading

If a greater understanding of PARROTDataLoader is needed, we suggest reading the Oasis report, as it describes how the database is constructed, which is what the methods found in PARROTDataLoader is based on.

The software for the PARROT application was presented in this chapter and the Source Code

for the features and the design has been described. Now that the application is in a working state and running, it is possible to make the tests of the Source Code.

CHAPTER 5

TESTING OF APPLICATION

In this chapter, we will look into the testing of the PARROT application, both through a usability test and through a general performance test.

5.1 Introduction

This chapter will document both our own testing phase as well as the usability test of PARROT, the latter performed in collaboration with all groups in the GIRAF Multi-Project. While the usability test has been introduced in the common part of the report(introduction chapter 1), the PARROT specific questions and results will be presented in this chapter. The testing phase was performed in combination with the PE-course Test and Verification, so the design was written early in the development of our system according to method in the book "Software Testing [9]" and the tests were performed after the development had been suspended. Because of this, there are tests which cannot be performed due to a lack of functionality.

5.2 Test Design Document

Identifier: PARROT#TD0100

Features to be tested

- Library navigation.
- Drag and drop pictograms onto the sentence board.
- Change the order pictograms appear in their library categories, using drag and drop.
- Check if these changes are saved.
- Audio functionality of pictograms.

- Playing sentences consisting of pictograms.
- Adding categories to the pictogram library.
- Adding pictograms to the library through the server/local database.
- Using the camera functionality.
- Storing pictures as pictograms in the library.
- Performance testing.
- Combining pictograms, and either parts of pictograms or other pictograms, to make new pictograms.
- Editing excisting pictograms.

Approach:

- Dynamic black box testing of the application on the target Android platform (Android 3.2).
- The Android tablet to be used for testing will be a Samsung Galaxy tablet 10.1 GT-P7510.
- If applicable, perform dynamic black box testing of the app on a later Android API (later than version 3.2) version on an Android Virtual Device.

Test case identification

- LibNav
- DragDrop
- AudPlay
- SentPlay
- AddCat
- AddPic
- AddCam
- Perform
- EditPic
- CombiPic

Pass/fail criteria

The app will have passed, if it manages to perform smoothly without any critical or serious errors, and with only minor cosmetic errors.

Identifiers	**Test item**	**Input specification**	**Output specification**	**Enviromental needs**	**Intercase dependencies (Depends on)**
LibNav	Navigating through the pictogram library.	Touchscreen input. Library input.	Being able to reach any part of the library.	Android tablet, Software testers. Library.	
DragDrop	Drag and drop pictures down to the sentences board.	Touchscreen input. Library input.	Correct consistent behavior. A pictogram is staying on the sentence board.	Android tablet. Software testers. Library.	LibNav
AudPlay	Playing the audio belonging to the pictogram	Touchscreen input. Pictogram with and without sound.	Playing the correct sound to the pictogram if there is a sound to the pictogram.	Android tablet, Software tester, Library.	DragDrop
SentPlay	Playning a sentences based on more than one pictogram.	Touchscreen input. Pictograms with sound	A sentences is read out as sound.	Android tablet, Software testers, pictograms.	AudPlay
AddCat	Adding new categories to the pictogram library	Touchscreen input. Data input to library.	A new category in the library with pictograms	Android tablet, Software testers, pictograms. Library.	LibNav
AddPic	Adding new pictogram to the library through a server	Touchscreen input. Input from a server	Some new pictogram in the library to use.	Android tablet, Software testers, pictograms. Library. External server.	
AddCam	Using the camera in the tablet with the app	Touchscreen. Picture from the camera.	A new picture taken by the camera in the tablet	Tablet with camera, Software testers.	AddPic
Perform	Testing that the app is performing smoothly and without performance issues	Touchscreen input.	A smooth test without problems.	Android tablet and our app.	All of the others.
EditPic	Making changes to the pictogram e.g. changing the text in the picture.	Touchscreen input. Pictogram.	A change in the pictogram .	Android tablet and our app.	
CombiPic	Combining pictograms and either parts of pictograms, or other pictograms, to make new pictograms.	Touchscreen input. Pictogram	Having a new pictogram to a new event.	Android tablet and our app.	

Table 5.1: Test Design

5.3 Test Result Document

LibNav

The Test was a success, we are able to scroll through individual categories aswell as change between several categories.

Result: Success.

DragDrop

The drag and drop functionality works with partial success. The application can successfully drag pictograms down onto the sentence board, as well as change their order on it. The application also remembers their order successfully if the user was to change between tabs or pause and resume the application.

However, in testing we discovered a critical error. On the speech board, if a pictogram is picked up from the pictogram grid and dropped off in the category list, the system crashes. This is not the case if you pick up a pictogram from the sentence board and drop it there.

Result: Partial Success.

Note: The discovered error can be fixed by removing an unintended feature from "BoxDragListener.java". This feature was intended to allow copying pictograms between categories, and was left in despite it having been discontinued.

Update: This error has been corrected by commenting out the code, so that it can be reimplemented correctly on a later date.

AudPlay

We were able to drag a number of pictograms down to the sentence-board and play their individual sound snippets.

Result: Success.

Note: Some delay was noted in the audio, however, after consulting with the voice actor, who is a member of the group, we have concluded that it is 'empty sound' in the recording and not a delay in the system.

SentPlay

This feature was not available for testing and will not be so before the deadline.

Result: N/A.

AddCat

We were able to create 108 categories without incident and decided not to create any more since we felt that this would be a realistic test for the capacity for categories.

Result: Success.

Note: After performing the test, we noted that putting the application on pause and resuming it resulted in a crash. Doing so again with only the two default categories did not. Doing so with two new categories deleted the new categories.

AddPic

This feature was not available for testing and will not be so before the deadline.

Result: N/A.

AddCam

This feature was not available for testing and will not be so before the deadline.

Result: N/A.

Perform

The application is smooth in performing its build in functions. However, startup takes a measured 10 seconds and does not contain a loading screen to inform you that the application is actually running.

Result: Partial Success.

EditPic

This feature was not available for testing and will not be so before the deadline.

Result: N/A.

CombiPic

This feature was not available for testing and will not be so before the deadline.

Result: N/A.

5.4 Usability Assignment

These are the PARROT assignments actually set at usability testing, they are based on the amount of functionality present in PARROT and that it should only take about five minutes in total to complete them.

1. Lav en billedrækkefølge af pictogrammer til "'Noah Nielsen"' i app'en/værktøjet "'PARROT"', der viser at han skal børste tænder og gå i bad, afspil derefter sætningen.
2. Opret en ny kategori.
3. Flyt de tre pictogrammer "'Sulten"', "'Tørstig"' og "'Søvnig"' til den nye kategori.
4. Giv den nye kategori et passende billede.
5. Giv derefter kategorien en behagelig grøn farve.
6. Tilpas "'Tale"' så kategori listen får en lys lilla farve, og giv sætnings brættet en mørk lilla farve.

5.5 Usability Test Results

The test results are noted in Table 5.2, they are the product of a group assembled by the Multi-Project Group. None of the member of the PARROT project participated in usability testing PARROT, so the severity of the errors are not defined by us, but by the persons running the test, as described in the IDA model.

5.6 Reflection

In our own test phase, we have been performing the tests in a manner most akin to dynamic blackbox testing. We have admittedly operated in a test to pass manner, but we have tested avenues of actions that we suspected could crash the system and did indeed do so. Also worth noting about our test phase is that while we are developers of the code, it is still blackbox testing as the test cases were written before the code was.
Based on our test design's Fail/Succeed criteria our application has failed its first testing phase. We experienced a critical error in the 'DragDrop' test as the system crashed, and a severe error in the 'Perform' test due to the poor handling of a slow application startup. Of course these things can easily be fixed, but should not be considered acceptable in a release candidate, due to our target group.

Our usability test was based on dynamic whitebox testing, as the questions were based on our knowledge of the code and the application.
Results from the usability tests were somewhat as expected, our design sadly isn't obvious. However, we never did intend for any user to go without some form of instruction. We do believe that it is an easy system to learn and remember, even if one might not be able to simply guess how it works.

Error	**Comments**	Total
Cosmetic	1. Could not move a picogram from box one to box two in the sentence-board 2. Could not supply a category with an Icon. 3. Lacked Confirmation when an Icon had been given to a category.	3
Severe	1. In PARROT the Back button did not give the expected result. 2. Participants could not understand that they needed to Long-click to move Pictograms. 3. The 'Color Palette' widget was difficult to use.	3
Critical	1. Participants could not figure out how to play sound and felt a button to do so was needed. 2. Participants could not create a new category. 3. Participants could not find the function to change speechboard color.	3

Table 5.2: Errors found in Usability Test

Cosmetic 1. This was a Design Choice, We felt that leaving open spaces between pictograms would be better, expecting that a user would take pictograms in order, or simply rearrange after filling it.

Cosmetic 3. This was an oversight on our part, we simply failed to consider that the need for confirmation. A mistake easily fixed by implementing informative Toasts (An Android widget.)

Severe 1. In the finished GIRAF system the return button was not supposed to be visible, so we did not handle it being pressed in PARROT.

Critical 1. The "play all" button was removed after a consultation with our contact (See Apendix C), a design choice that might need to reconsidered. This could be remedied by bringing back "play all" to the design, or adding smaller play buttons under each pictogram in the sentence board.

In this chapter, we presented the testing material for PARROT, and examined the results for ideas on how close our application was to the idea in the design phase, and what we could do to bring it closer to the ideal.

CHAPTER 6

DISCUSSION

This semester we have been working on creating an Android application for children with ASD. This has brought many new and interesting aspects we have not experienced in former projects. The idea of working in a multi project consisting of many groups that need to work together is completely new to us. Creating a project we know will be used in real life at some point is new as well, as is the agile development method we have been using. Finally we have not been working with Android applications before, which have been rather interesting.

The multi project

This semester has differed from other semesters drastically by having a multi project instead of a smaller individual project. This have had a large impact on how the groups have had to work, and made it necessary to cope with problems that we have not encountered before.
The groups are largely dependent on each other as all parts of the product needs to work together. We realized this early on, and have taken steps to increase communication between the individual groups. One of these steps were to hold weekly meetings, where we presented what we had been working on, as well as what we where going to work on.
This helped us to constantly know where everyone was at any given time, and helped communication between the groups greatly. At this meeting we also shared knowledge between groups, so that problems one group could have run in to could be avoided by the other groups.
Another step that has been taken is to eat together in the cafeteria. This have greatly improved communication between groups. Here we could talk about our free time, but we also talked about the projects more common aspects and helped communication on a daily basis without it being too formal.
The dependency of other groups' work have also had a great impact on our work process. The good thing is that it has become far easier to specialize on one field. For instance we have used

a database, but needed to create and maintain it, only ask the group handling the database for functionality. One of the problems, though, are that this also means that decisions can be made in the multi project or other groups instead of the individual groups.
The individual groups need to accustom themselves to the fact that they might need to adjust their ideas to that of the multi project, or another group. For instance have we in our group wanted some extra tables in the database, but the multi project decided against this.
We believe this way of working gives a better view on how work is done in the real world. The way we have been working this semester feels like working in a bigger corporation rather than in small individual project groups. The fact that the group rooms are shared by two groups only separated by cardboard stands resemble cubicle workstations, which have had decrease in efficiency of brainstorms and discussions, but increased communication across groups.

Agile method - SCRUM:

Another new experience in this project is that we chose to work with a branch of development methods that we have not work with before. This new branch of development methods is the Agile paradigm, and the method that we have chosen is SCRUM. Agile methods promote adaptivity, and the idea of changing the plans on the way, working in iterative steps in close contact with the customer, analyzing, implementing and testing along the way. It also incorporates the idea of creating one functionality at a time, and adding it to the complete system as it is developed. This means that it has been possible for us to make a list of functionalities, and then focus on implementing them one at a time.
We have not implemented them all, but we never expected to be able to do that in the given time frame. Instead we have made a complete list of functionality that, if implemented, could complete the PARROT application so that it would be ready for deployment. We hope that future project groups that continue the PARROT project will take this list to heart, and complete the PARROT application.

The SCRUM method involves programming in short-term session known as sprints. In these sprints we program the features, which are planned immediately before the sprint during the sprint planning meeting.
How much effort and time we use for any feature is determined by utilizing planning poker, which means that every member of the PARROT group is given a deck of cards. When a feature is listed, each member chooses a card, representing the member's estimate of the amount of work needed for this feature. We do this because it helps us organizing a sprint and keeping realistic goals, all within a short amount of time.
In our work process, this has helped keeping a steady stream of work throughout the whole semester, and giving the group a constant feeling of achieving goals.
To make sure that we know what functionalities are being worked on, how far we are currently, and generally keep track of our progress, we have assigned a SCRUM master. This person have tried throughout the semester to update a burndown chart of our progress, and copy it out to the multi project bulletin board where all burndown charts are pinned.

This chart also helps the group to know how far we are, and have often given the group a morale boost.
We have added a few new procedures to the SCRUM method. Instead of a stand up meeting, we have used walk and talk. This has helped us have a fresh mind on subjects, and given us fresh air while talking.
In addition we have added the procedure of Pair programming, which have helped correct mistakes while writing, and have increased performance and quality overall. A last procedure we have added is a board, divided into two parts: “Free” and “In Use”. In these two parts, multiple sticky notes are shown, each representing a class or an xml file. When one is being modified, a note is moved from the Free section to the In Use section under the name of the programmer that is using it. This has helped us avoid revision conflicts.

Real World Assignment

As opposed to the last semester we have been working on a project that, if luck has it, will end up in the real world.
We have had contact with institutions through contacts that have described the needs of the institution.
This have had a great impact on our work, as we know that one day the project might be used by an institution which helps children with ASD. We have had some interviews with these contacts, as well as having been on a visit to the institution to get a better insight of in what context PARROT will be used. By this, we have learned the importance of handling pictograms for the children, and we learned that we needed to make sure that it could be used on a simple level, rather than with a lot of fancy features.

Android

Working with Android development has been an interesting challenge. First of all, Android incorporates the way of creating a user interface in an easy way, primarily because of the way the graphical components are created in XML.
In comparison with the other projects that the group has taken part of, this project focuses more on creating a graphical user interface, that is functional and safe to use.
To make the design simpler and more organized, we split the design into tabs, each refering to an individual fragment of the functionality. We felt that this was smart, since we could restrict access to the individual tabs, thus ensuring that the children would only have access to the functionality they would need.
Since we were using Android, we would be working with a touch screen input, which required us to rethink our usual design approach, since we were used to having a mouse and keyboard as input.
Finally, working with new technology was interesting, and gave us insight into Android development which we can use for personal projects.

CHAPTER 7

CONCLUSION

The problem definition states:

> *How can we ease the daily life for children with ASD and their guardians, while complying with the study regulation?*

We have designed a product that should help ease the use of pictograms. PARROT should help organize and ease access and use of pictograms. This allows for many of these to be handled simultaneously, without the need of the many physical pictograms and their bulky folders as mentioned in the analysis.
It also gives the functionality to play sound, which could help some of the children learn the spoken language while using the pictograms to communicate. The idea of handling pictograms digitally also opens opportunities otherwise not easily accessible for the guardians. For instance, we can change PARROT's colors according to the needs of the individual child. This includes the colors for the different categories as well as those of the speech board. The guardian can modify the categories and organize them according to the needs of the child.
Since we have developed an application based on the needs of a third party, have been part of a multi project, and have performed dynamic blackbox testing as well as whitebox usability testing, we have upheld the goals of the study regulation.

The whole project has been part of a greater multi project with five groups participating. As such we have not made an individual product, but one that is part of a greater product, GIRAF. GIRAF is not yet finished, but is hopefully going to be continued by other students on future semesters, so that it can become a product worthy of being used in the aforementioned institutions.
As part of the multi project we have been using an agile development method, SCRUM. Using SCRUM we have not spent the start of the semester analyzing the whole system, and designing it from the start. Instead we have analyzed and designed parts of the system one sprint at the time. In previous semesters, we have wasted time on analyzing functionality that was never

implemented, which was not the case in this semester.

CHAPTER 8

FUTURE WORK

After the implementation of the application PARROT based on the design mentioned in chapter 3 has stopped, and the test of the application and the usability test have been performed, we have a list of bugs and unimplemented features.
In this chapter we want to go through the list of bugs and unimplemented features for the sake of further development of this application.

All tasks involving the bugs, unimplemented features, and other deficiencies, can be split up into three groups:

The first group contains the bugs found in the tests and features that are necessary for the application to work properly with the other parts of GIRAF.

These are:

- **Fix the bug that causes the system to crash when the user has no categories:** In the application the user can delete categories in the Administration tab, but in the tests a bug occurs when the user deletes all the categories. This error makes the application crash, so this is a bug that needs to be fixed so that if the user deletes all the categories, the application does not crash.
- **Fix the bug that causes the system to crash when a category is copied into itself:** In the system it is possible to copy a pictogram form one category into another category, and copy a whole category into another category. But if the user tries to copy a category into itself, the system crashes. This bug needs to be fixed so that if the user by accident makes this mistake the application will not not crash.
- **Finish connecting the application to the launcher:** When opening the PARROT application, the application should get a profile from the launcher, so that the application

knows which pictograms the profiles has, and other options. This was not finished in time, so in our test the profile we are using is a “dummy profile ” so for the application to work, the connection between the launcher and the PARROT application needs to be fixed.

- **Remove the DragShadow when dragging an empty Pictogram:** Currently if the user drags one of the empty pictograms in the sentence board, the shadow will be shown. This shadow must be removed to prevent confusion.
- **Move all save calls to the PARROTDataLoader to the onPause method of PARROTActivity:** In the current version of PARROT there are save call in both onPause of PARROTActivity and ManageCategoryFragment. This should be fixed, so that only PARROTActivity has a save method in its onPause method. Note that the individual tabs should still be able to make save calls, they should just not do it automatically.

The second group contains the unfinished features, meaning features that we have commenced implementing but not finish due to a lack of time.
These features are:

- **Make the Spinner in the administrations tab work, so that it contains all the child profiles for the active guardian:** The spinner(Android equivalent to drop down menus) is not finished, so is does not show all the profiles that the guardian have access to when the guardian is in the administration tab.
- **Show and Edit Category names:** In the application we are not yet done implementing the category’s name in the user interface. Also we are not able to edit the names in the current state.
- **Add guardian mode in child mode:** At the moment the users, children as well as guardians, can access all the tabs in the application. This is not the intention. The meaning of the guardian mode is that the children can only access the speech board tab and to access the others tabs the guardian needs to authorize with their QR-code.
- **Note that the name of the root folder might not be sdcard/ on all Android units.**
  - **Consider using Enviroment.get(...) to make the paths hardware independent.**
- **Add Profile icon to design.** The original design of the speech board had a pictogram of the child in the upper left corner. This has yet to be implemented. Our idea was that this pictogram could be used in the sentence board just like the other pictograms.
- **Write the functionality for the unimplemented buttons on the ManageCategoryFragment:** In the administrations tab there are some buttons that yet has no function implemented due to the lack of time.
- **Colored edge on the categories:** In the application we want to have a colored edge around the categories’ pictograms to show which category is connected to the displayed pictograms.

The third group contains the features that have not yet been implemented.
These features are:

- **Add camera functionality and the ability to add the pictures taken by the camera to the PARROT application.**
- **Suggest to the Admin group that Pictograms and Categories are made part of the database:** We felt that this would be a good idea since pictogram is a important part of the GIRAF applications and making them a part of the database will make it easier to share among the applications.
- **Expand the options fragment, so that the children can be more individualized.**
  - **For instance, add functionality to limit the access to the individual tabs.**
  - **Remember to save all changes to the PARROTProfile:** Remember to update the PARROTProfile class to handle the settings.
- **Check with customers/experts if the drag and drop functionality with sentences are handled correctly.**
  - **Or make it possible to change how the sentence board functionality should work for the individual child.**
- **Consider changing the Category class, so that categories can contain categories.**
- **Refactor BoxDragListner to SpeechDragListner:** This serves the purpose of organizing the code in a better way.
- **Create a startup screen while loading:** In the current state the application takes a while to startup and this confused and annoyed the test subjects. Adding a startup screen would inform the user that the application has started.
- **Create a busy animation:** This would be just like the hour glass in Windows.
  - **This one could be common for all applications for the GIRAF platform.**
- **Handle colors from the launcher and make those standard colors in PARROT.**
- **Write search functionality for pictograms, and use tags:** This will be needed when the application will get the full database of pictograms and the guardians need to find a specific pictogram.
- **Add mute functionality:** Some of the children do not respond well to sound.
- **Option of displaying the categories in speechboard the same way as those in the management tab.**

We hope that this chapter have provided future developers with the idea of where to starts when continuing on the PARROT project. We feel that we have presented the problems and shortcomings that we have found within the PARROT application, which has to be resolved in order to ensure a fully functional application to fit the current needs of the costumer.

APPENDIX A

NOTES FROM INTERVIEW

This is notes from an interview with Mette Als Andreasen, an educator at Birken in Langholt, Denmark.

Når tiden løber ud (kristian har tage et billede):
Færdig - symbol
Gå til skema - symbol
Taget fra boardmaker

Kunne være godt hvis man kunne sætte egne billeder ind som start/stop symboler.

Rød farve = nej, stop, aflyst.

De har sådan et ur på 60 minutter hvor tid tilbage er markeret med rød, og så bipper den lige kort når den er færdig.
Det ville være fint hvis de kunne bruge sort/hvid til dem der ikke kan håndtere farver, men også kan vælge farver.

Stop-ur:
en fast timer på 60 minutter + en customizable som ikke ser helt magen til ud, som f.eks, kan være på 5, 10 eller 15 minutter for en hel cirkel.

timeglas:
skift farve på timeglassene, men ikke nødvendigvis gøre dem større. Kombinere med mere/-mindre sand. Eventuelt kombinere med et lille digitalt ur, til dem der har brug for det, skal kunne slåes til og fra.

Dags-plan:

ikke særlig relevant til de helt små og ikke særligt velfungerende børn. Men kunne være rigtig godt til de lidt ældre.
En plan går oppefra og ned, og hvis der så skal specificeres noget ud til aktiviteterne, så er det fra venstre mod højre ud fra det nedadgående skema.

Til parrot:
Godt med rigtige billeder af tingene, som pædagogerne selv kan tage, eventuelt også af aktiviteter, så pedagogerne kan have billeder af aktiviter som de kan liste efter skeamet.

Der var mange skemaer rundt omkring, og der henviser det sidste billede i rækken til næste skema, som hænger f.eks. på badeværelset eller i garderoben.

APPENDIX B

INVITATION TO USABILITY TEST

Kære ekspert

Vi vil gerne inviterer dig til at deltage i den første brugervenligheds test af GIRAF, en Android applikation bygget til børn med autisme. Formålet med denne test vil være at undersøge hvor brugervenlig applikationen er og hvor nemt eller svært det er at bruge den. Derfor er det helt fint hvis du aldrig har set eller hørt om denne applikation før nu, da vi gerne vil observerer, hvordan første gangs brugere så vel som brugere med kendskab til applikationen, har det med denne applikationen.

Bemærk venligst at vi er ikke tester din kendskab til applikationen eller evner med en tablet, men derimod om GIRAF applikationen er nem at bruge, vi har kun interesse i at kende til de svagheder der ville være i applikationen. Dette betyder også at du ikke kan give nogle forkerte svar, da du er eksperten.

Derfor vil vi gerne inviterer dig ud i vores brugervenligheds laboratorie, hvor vi kan studere din brug af applikationen. Under brugervenligheds testen vil du blive givet en række opgaver, som skal udføres. Yderligere vil du blive bedt om at tænke højt og fortælle alle tanker, indtryk og valg du tager ved brug af applikationen under testen. Under testen af applikationen vil der blive optaget både video og lyd, til at studere testen senere.

Dagen kommer til at bestå af:

Vi vil meget gerne høre fra dig hvis du har lyst og tid til at deltage i denne brugervenligheds test, den 22/5 - 2012, på Aalborg Universitet.
For at vide hvornår på dagen du kan komme vil vi gerne, at du går ind på denne side (http://www.doodle.com/d2h6swgbtsdf6z2b) skriver dit navn og vælger det tidspunkt på dagen du helst vil komme, dette er svar nok for at vi ved du gerne vil komme.

Kommentarer og spørgsmål kan sendes retur til den mail invitationen kom fra.

På forhånd tak,
Android projektet
Software 6. semester
Aalborg Universitet
Selma Lagerlöfs vej 300, 9220 Aalborg

AALBORG UNIVERSITY

Figure B.1: Invitation sent to the test persons of the usability test.

APPENDIX C

NOTES FROM INTERVIEW WITH TOVE

Dagsorden:

- Præsenter os selv, og hvad vi laver.
- Use cases:
  - Daglig brug af pictogrammer.
    - En tablet pr. barn
      - Nogle vil kun have et billede pr. tavle.
      - Nogle kan håndtere flere billeder.
      - Nogle kan håndtere en dag
      - Nogle kan håndtere en hel uge. Med brug af farvekoder.
    - Jeg vil gerne... den slags ord skal også være med.
    - Forberedelse til hvad de skal gøre/foretage sig.
    - Meget individuelt
    - Egne kategorier er en fin ide, men de har også generelle kategorier
- Præsenter vores prototype.
  - Modtag konstruktiv kritik.
- Evner, hvad kan de?
  - Kan de forstå undermenuer/underkategorier?
    - Har de behov for dem?
  - Vil det være en god ide at kunne gemme sætninger, eller skal de dannes igen hver gang?
    - Vil det kunne variere fra person til person?
      - Ja, det varierer.
- Sidste punkt: Skift af kontaktgruppe.

Spørgsmål:

- Hvordan skal tekst placeres i forhold til billeder?
  - Over, under, ovenpå...?

Noter fra tove

Spil med brug af stemme.
Stimulation af kommunikation.
Hvert ord skal afspilles individulet, og skal have højere prioritet end at kunne afspille en sætning.
Billede af barnet i øverste hjørne skal med.
Skal også kunne bruges som pictogram
Tekst må gerne kunne skjules.
Underkategorier kan måske være en fordel.

Der skal kunne være mulighed for at lave et ”board” af elementer, som de skal kunne vælge ud fra, frem for kategori listen.
Kan eventuelt laves som en midlertidig kategori lavet af en guardian.

Mulighed for at håndtere valg mellem to (evt. flere) ting, f.eks. ved kun at vise de to elementer på skærmen. (Håndteres af guardians sammen med et barn)

Billeder fra hverdag, med lyde.
Visuelt feedback for at gøre noget rigtigt/forkert.

APPENDIX D

BRIEFING, DEBRIEFING AND QUESTIONNAIRES FROM USABILITY

Briefing

Goddag og velkommen til denne brugervenlighedsundersøgelse.

Vi vil gerne starte med at takke dig for, at du vil hjælpe os med at gennemføre denne brugervenlighedsundersøgelse. Vi læser op fra dette dokument for at sikre os, at alle personer som deltager i vores studie for samme introduktion. Hvis du har spørgsmål undervejs, er du naturligvis meget velkommen til at stille disse spørgsmål.

Vi har i dette semester bygget et system til Android til at hjælpe børn med autisme og deres pædagoger og forældre, og det er nu nået til et stadie hvor vi gerne vil teste systemet. Denne test handler udelukkende om at finde problemer og mangler i systemet, og ikke om at teste jeres viden af systemet, så alle tanker I må have om produktet vil vi meget gerne høre.

Før vi starter første del af testen, vil jeg bede dig om at underskrive denne samtykkeerklæring for at sikre, at du er indforstået med rammerne for studiet. Derudover skal du også svare på et demografisk spørgeskema inden testen går i gang.

Testen består af fire dele:

- Test af applikationer (20 min)
- De-briefing og spørgeskema (5 min)
- Test af Administrations applikation og web applikation (20 min)
- De-briefing og spørgeskema (5 min)

Undervejs vil der være en pause.

I de to tests vil du blive stillet en række opgaver som du skal løse. Læs opgaveformuleringen grundigt og fortæl så test hjælperen hvad du mener opgaven går ud på. Derefter skal du forsøge at løse opgaven så godt som muligt. Opgaverne skal løses i den rækkefølge de står således at du starter med opgave 1 og arbejder dig ned af.

Det er meningen at du skal tænke højt mens du løser opgaverne. Dvs. at du siger hvad du har tænkt dig at gøre for at løse opgaven, hvilke ting du synes virker uklare eller komplicerede og hvordan du tror systemet virker. For eksempel vil det være godt hvis du nævner hvad du forventer en knap gør inden du trykker på den.

Når testen er færdig vil der være nogle afsluttende spørgsmål som du skal besvare omkring hvordan du synes testen er forløbet og hvad din opfattelse af systemet er.

Figure D.1: Briefing for test subjects

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

__ Kvinde X Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
X Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

X Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Parrot:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Wombat:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

X Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
X Lettere erfaren
__ Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

X Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Parrot:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Wombat:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

X Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
X Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

__ Ja X Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
__ Let
__ Middel
X Svær
__ Meget svær

Parrot:

__ Meget let
__ Let
__ Middel
X Svær
__ Meget svær

Wombat:

__ Meget let
X Let
__ Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

X Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
X Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

X Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
X Let
__ Middel
__ Svær
__ Meget svær

Parrot:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Wombat:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

X Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
X Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

X Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Parrot:

__ Meget let
X Let
__ Middel
__ Svær
__ Meget svær

Wombat:

X Meget let
__ Let
__ Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
X Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
__ Middel
X Svær
__ Meget svær

APPENDIX E

SETUP AND INSTALLATION

This section will cover the procedures that a future developer will have to go through in order to get the current version of PARROT up and running. We will go through how to install the application on a Android tablet, and how to get the PARROT project ready for development in Eclipse

E.0.1 Installing the Application

In order to install the current version of PARROT on an Android tablet, these steps must first be completed:

- The tablet must support Android version 3.2, otherwise PARROT will not run.
- (Optional) Install Launcher_004.apk from the GIRAF launcher group on the tablet.
- LocalDB.apk from the Admin group must be installed on the tablet.
- The folder Pictogram and all that it contains must be copied to /root/Pictogram on the tablet.

When all of the above steps are successfully completed, the current version of PARROT.apk can be installed on the tablet. Installing the launcher is currently optional, but we recommend that it is done, since GIRAF is meant as a packaged project, rather than a number of completely individual applications.

E.0.2 Setting up Eclipse for developing PARROT

While installing the application can give a quick view into how it is structured, the source code needs to be accessed in order to make changes to the application. In order to help future developers get through this phase without too much trouble, we have written a short list of instructions on how to do this:

- Make sure that Eclipse is installed on your computer (Indigo version or newer).
- Install the Android ADT plug-in for eclipse http://developer.android.com/sdk/eclipse-adt.html#installing
- Make sure that the tools for Android version 3.2 is downloaded.
- Go to the tags/project/PARROT folder on SVN and import PARROT and AmbilWarna.
- Go to properties>Android for PARROT, and check that the build target is 3.2.

  While there, make sure that AmbilWarna is included as a library.
- Now go to Properties>Java Build Path and make sure that oasislib.jar is there. If not, press **Add JAR's** and add it from there.

  Now the project should be able to compile. Note that in order to run the application, the Android devide or Android Virtual Device needs to be set up according to the description above this.

APPENDIX F

LIFECYCLE OF AN ANDROID APP'S ACTIVITY

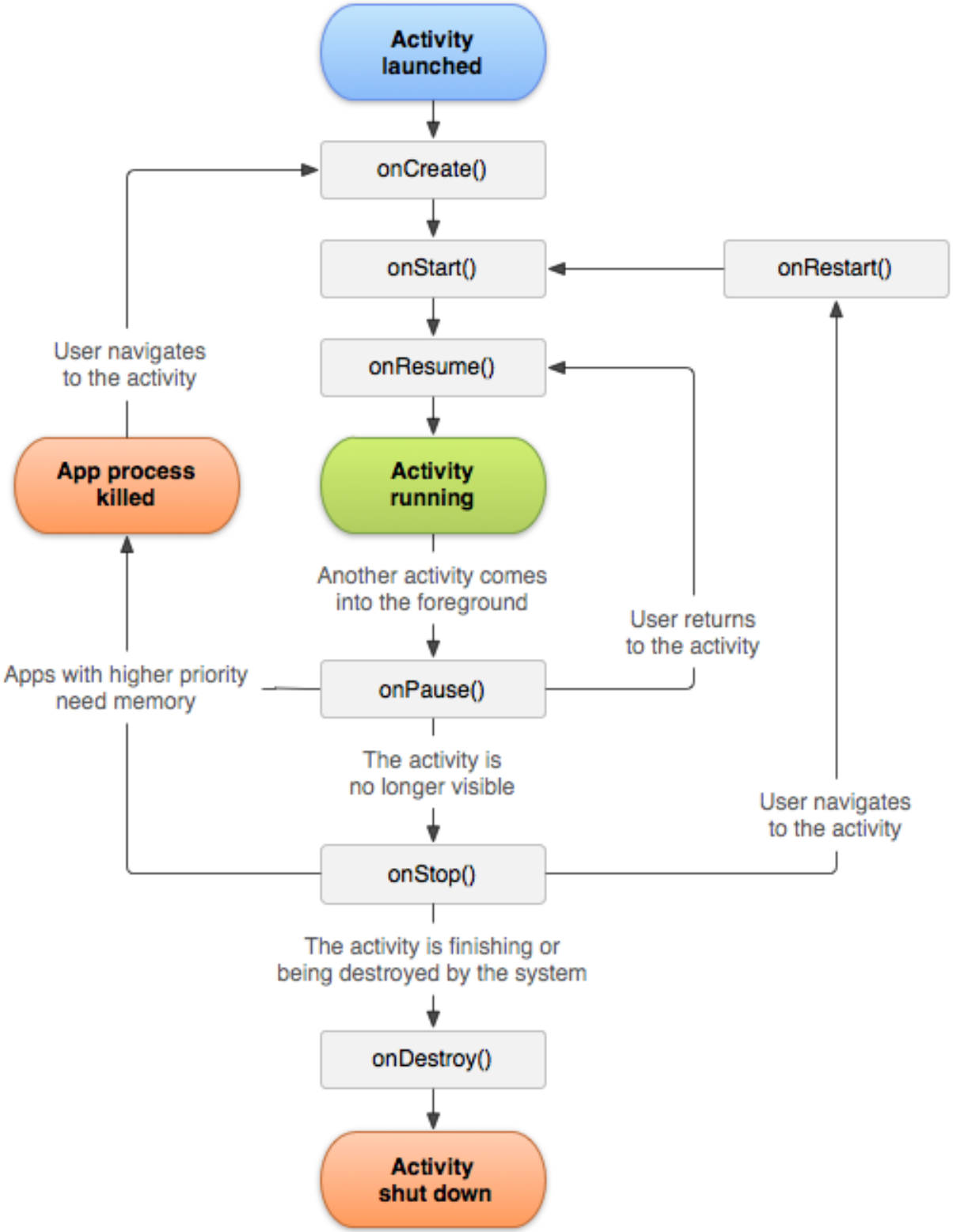

Figure F.1: The LifeCyle of an Android App's Activity.[12]

APPENDIX G

ERRATA

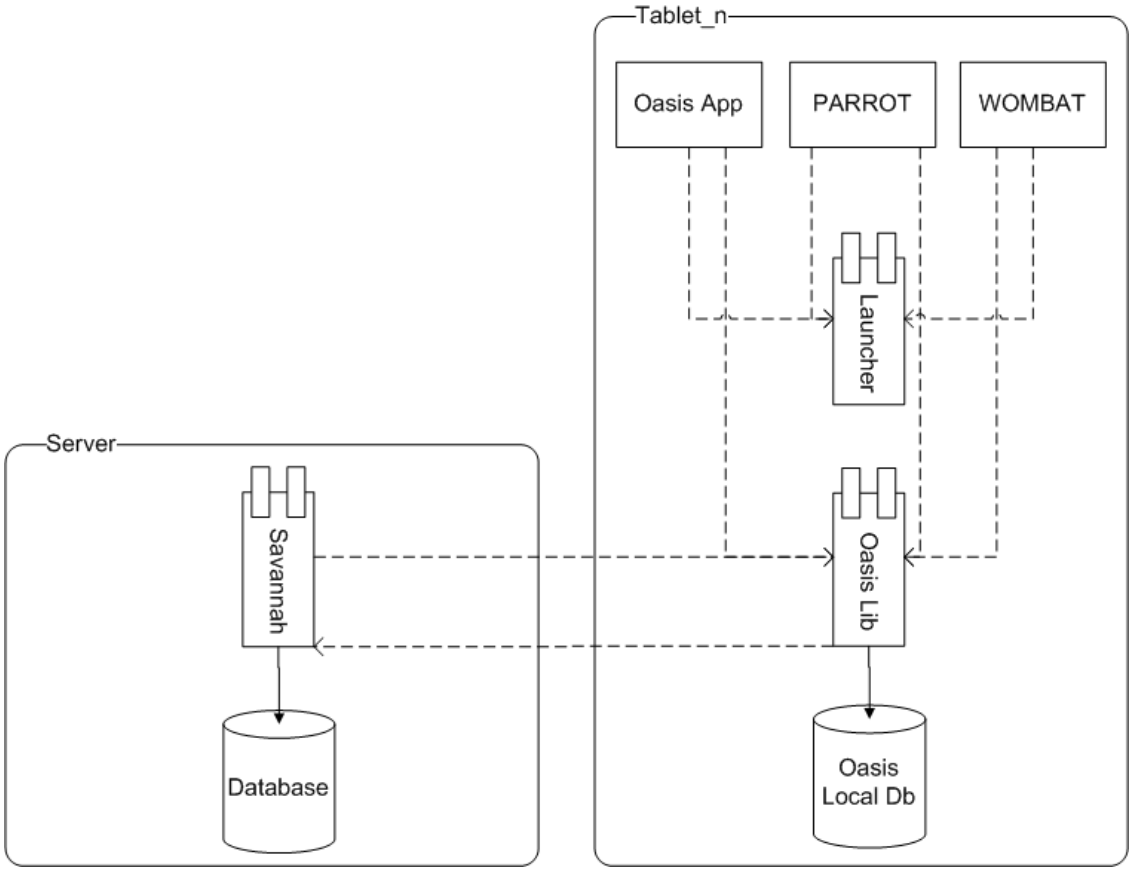

Figure G.1: An Erratum of the GIRAF Component seen in Figure 1.1.

BIBLIOGRAPHY

[1] Hello gridview. http://developer.android.com/resources/tutorials/views/hello-gridview.html.

[2] Scrum Alliance. Advice on conducting the scrum of scrums meeting. http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting, 2011.

[3] Scrum Alliance. Scrum alliance. http://www.scrumalliance.org/, 2011.

[4] Matthias Kaeppler Charlie Collins, Michael D. Galpin. *Android in Practice*. Last viewed: 2012-05-30.

[5] Temple Grandin. Teaching tips for children and adults with autism. http://www.autism.com/ind_teaching_tips.asp, December 2002.

[6] Steffan Bo Pallesen Jacob Bang, Lasse Linnerup Christiansen. Administration module for giraf. *http://people.cs.aau.dk/~ulrik/Giraf/Admin.pdf*. Last viewed: May 2012.

[7] Steffan Bo Pallesen Jacob Bang, Lasse Linnerup Christiansen. Administration module for giraf. *http://people.cs.aau.dk/~ulrik/Giraf/DigiPECS.pdf*. Last viewed: May 2012.

[8] Jan Stage Jesper Kjeldskov, Mikael B. Skov. *Instant Data Analysis: Conducting Usability Evaluations in a Day*. Last viewed: 2012-05-24.

[9] Ron Patton. *Software Testing*.

[10] Samsung. Samsung tablet. test http://www.samsung.com/global/microsite/galaxytab/10.1/index.html.

[11] Yuku Sugianto. Ambilwarna. http://code.google.com/p/android-color-picker/, 2011. Last viewed: 2-6-2012.

[12] Android Development Team. Activity life cycle. http://developer.android.com/guide/topics/fundamentals/activities.html, 2012.

[13] Android Development Team. Android actionbar guide. http://developer.android.com/guide/topics/ui/actionbar.html, 2012.

[14] Android Development Team. Android fragment guide. http://developer.android.com/guide/topics/fundamentals/fragments.html, 2012.

[15] Android Development Team. Android layout declaration. http://developer.android.com/guide/topics/ui/declaring-layout.html, 2012.

[16] Android Development Team. Android type guide. http://developer.android.com/guide/topics/resources/more-resources.html, March 2012.

[17] Aalborg University. Studieordning for bacheloruddannelsen i software. *URL: http://www.sict.aau.dk/digitalAssets/3/3331_softwbach_sept2009.pdf*. Last viewed: 2012-03-15.

[18] Don Wells. extreme programming, 2009. URL: http://www.extremeprogramming.org/rules.html.