PARROT

Spring semester 2012

Group: sw601f12

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

Study Report

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

Title: Android

Subject: Application Development

Semester:
SW6, Spring Semester 2012

Project group:
sw501e11

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

Supervisor:
?

Number of copies: ?

Number of pages: ?

Number of appendices: ?

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 based application for children with Autism Spectrum Disorders. The first part of the report is a common part, which presents the goal for 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. Last a test chapter will be present what we have achieved in this semester, and conclude what worked and what did not work, in order to determine what to put in the future work section.
This report is also written as an aid for the next years 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 a application in android to android based tablets.

The reader is expected to have some basic knowledge about programming application 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 alphabetic letters followed by a number, like this; [**?**]. Sometimes it is followed by a comment used to precise the reference, like this; [**?** , 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.

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.

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

INDHOLD

KAPITEL 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[6]. 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 [3], 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[5]. 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. [**?**]

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) [7], and Scrum [2].
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 [1].

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

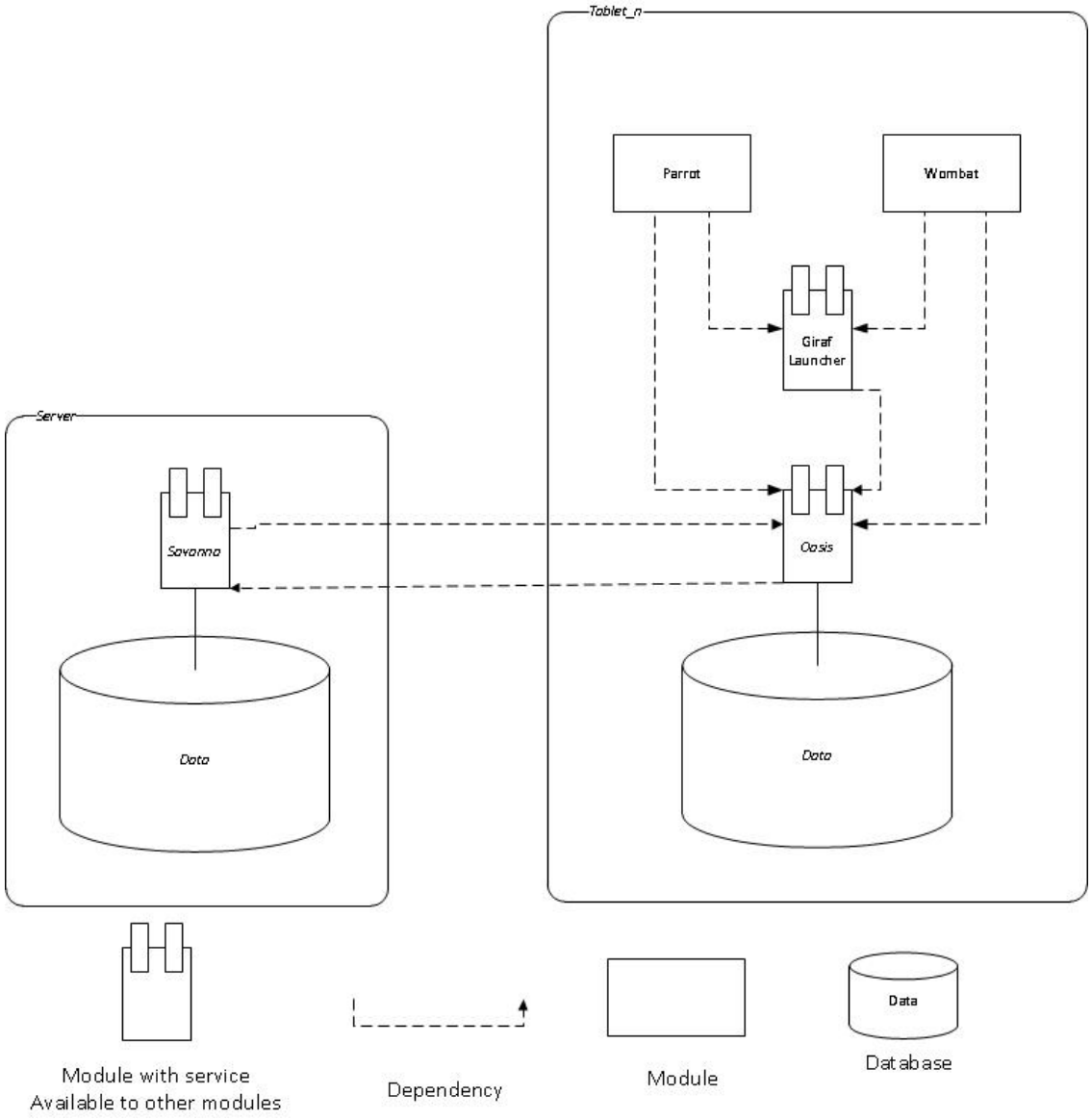
Our System architecture – shown in Figure 1.1 has been designed with simplicity in mind and was greatly inspired by the MVC pattern. This means that the architecture is divided into three layers. The lowest layer is the database where the information is stored. Above this layer is the controller layer 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 [**?**] 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 to 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 for the test is the five contact persons. We assess that they, as a test group, are representative. We base this on them being a mix of pedagogues and teachers, with varying computer skills.
Even if they have some 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 Figure B.1.



Figur 1.1: The GIRAF architecture

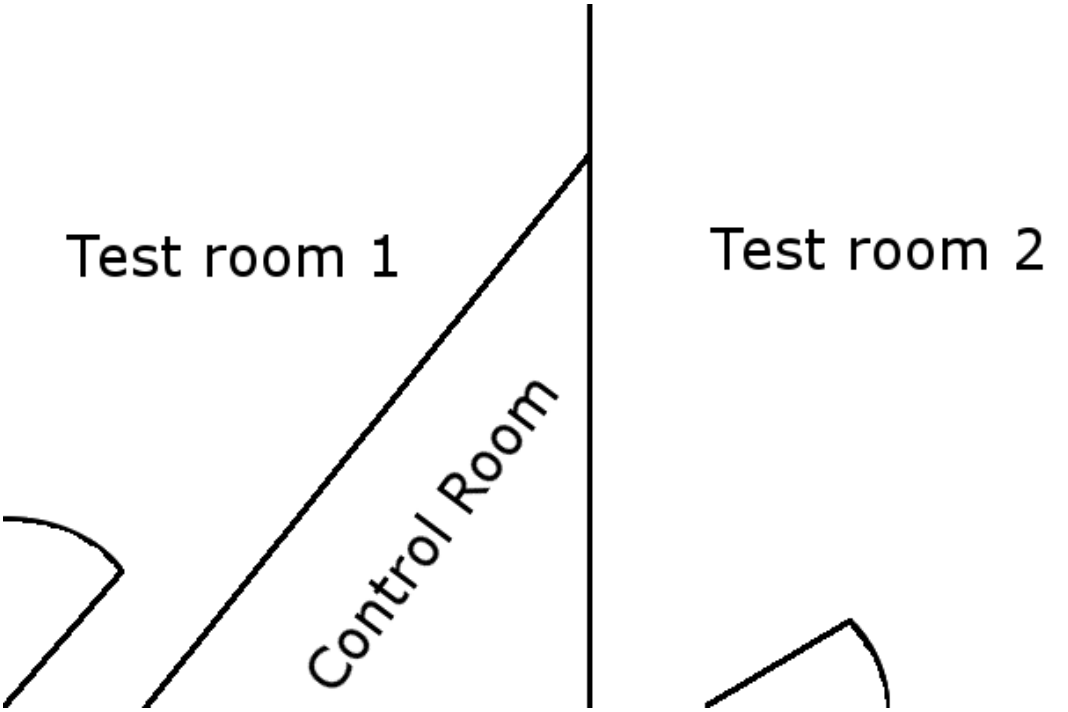
Based on the fact that the test should be short and the test group is small, the Instant Data Analysis (IDA) method for usability was chosen. [4]
A traditional video analysis method could have been used, but was not estimated to be time-effective enough to be chosen.

Setup

The usability test is divided into two tests: a test of the three applications, and a test of the two administration applications – tablet application and 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

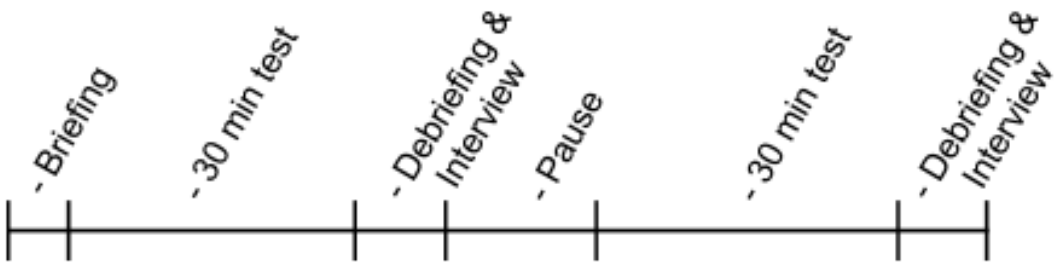
The usability lab on Aalborg University is designed with two rooms for usability testing and a control room to observe and record the tests. The two test chambers were assigned a test each and the control room were used to observe both tests as seen in figure 1.2.



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

As an precaution, all tests were recorded on video and audio.

Execution



Figur 1.3: The schedule of the usability test.

The tests were conducted according to the schedule in Figure 1.3.

Briefing, debriefing, and questionnaire documents can be found in **??**, and the results of the test can be found in **??**.

KAPITEL 2

ANALYSIS

To get a greater knowledge of how to work with children with autism and which tools are in use in the daily work with the children, we has been in contact with Tove Søby, a speech therapy consultant that works with children with autism.

Based on the meeting with Tove **??**, we have learned that one of the tools used in the communication with children with autism is pictograms. The pictograms are used in a schema for the day, where all their daily activities are listed as pictures. This gives the children the possibility to go to their schema and see what thay are going to do next. The pictograms are also used for direct cummunication with the children. The pictograms are used by the children to contruct sentences to communicate with others, and also to try to teach the to speak if that is a problem for the specific child.

In the use of pictogram, the problem is that they are very space-occupying, and the tools are not very practical to move around, since they consists of a bookcase full of folders with pictograms. Another problem is that when new pictograms are needed, the personel that needs the new pictogram has they print them out, cut them out in small squares, and laminate them. So in adition 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 is 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 the no children are the same, as mention in the introduction,`!sæt ref til 1.2 target group!`. 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 ovn 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 samne functionality, just converted to a digital version.

KAPITEL 3

APPLICATION DESIGN

In this chapter we will present the design for our android application and short describe which feature the application will contain, what the d and why we have choses these features to the android application.

Based on the original 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.
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 user should not (commonly) 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 because the application is meant to be used by children with autism. 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. It is not our thought that the system has to be intuitive, but it is our opinion that if the user is provided by 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, such 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, so we drew up a quick sketch of how the system would look in our opinion.
As seen in the picture, we wanted a speech board to show the current available pictograms,

Figur 3.1: Sketch of the speech board tab

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 existing tools as possible. The guardians already have speechboards 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 sentenceboard 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 buttons, 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 of the speech board, we continued on to the parts of the application that is restricted to the guardians. We had to make the design so that the options and the administrations 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 what the application from last year did but this was not possible so we began to search for other ways to do so. We came up with 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 see figure 3.2. First we have a spinner 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 pictogram that is found in the selected category.
Now we had been through the design for the application, we also have a lot of features for the application:

- **Drag and Drop:** This feature allows the user to choose a pictogram and 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.

Figur 3.2: Sketch of admin for categories

- **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 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 main points in our application is that we can adjust the application to needs of each child.
- **Showing Pictogram:** This feature takes the pitograms that is a part of the users profile and put it into the pictogram grid in the speech board tab. This feature is important for 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 that is it all this application is about.
- **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 see figure 3.2 we contacted Tove to get some feedback for the designs and she gave a positive feedback so we put our design in to production.

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 imported they are for the end product.

KAPITEL 4

IMPLEMENTATION

In the following chapter the software 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.

4.1 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:

We do this by creating listeners to attach to the views we want to drag from or to. These listeners then stores information and listens for different actions, which include DRAG_STARTED and DROP which helps us define different behavior. Drag and drop is simply but 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.

Execution:

So far PARROT have 2 tabs which include drag and drop. To optimize the code we have created a listener for each of these tabs. The tabs holds a few information's that the listeners utilize. These include 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 are to be performed.

An example on that can be seen in codesnippet REFFER TO CODESNIPPET that show the variables from SpeechBoardFragment.java. These are first initialized to -1 to make sure no mistakes are made.
In the tabs we also select what views should handle drag and drop. The listeners run for all views for the tab simultaneous. When a object is dropped, all of the handled views will notice the DROP action. Therefor we have a boolean that handles if the object being dropped is inside a specific view. If the object leave the view this is set to false, and if it enters it is set to true, which can be seen in codesnippet REFFER TO CODE HERE. 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.
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 are the view that are currently being handled. What view the object is from is gathered from the tab via the former mentioned information. With these two information we know where the object is from, and where it has been dropped, and therefor we know what action to perform. These actions vary and some will be described in other sections. REFFER TO THOSE OTHER FUNCTION SECTIONS 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 and 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 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:

We have used the book Android In Practice REFFER TO BOOK HERE. Good guide on how drag and drop functions. ADD ALL FUNCTIONALITY THAT IS IN THE TWO DND CLASSES

4.2 Playing Sound

4.2.1 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 **INSERT REF!!!!!!!!!**, the reason

for this being 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.

4.2.2 Solution

Writing the Pictogram class so that it includes functionality for playing its asociated 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 thoughout the entire app very easy.

4.2.3 Execution

To use the a 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 are handled in two different places in cooperation, the Pictogram class and the AudioPlayer class.
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 reponsive 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 4.1: test

For instance, suppose we want to play the word of a pictogram (The word is a sound file with the pronounciation of the pictogram). The first thing we do is call the **playWord** method form the Pictogram in question. This method cheks 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 the AudioPlayer class, more specifically 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  }
```

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.

4.2.4 Result

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

4.2.5 Notes:

At the current state of the program, the only place in which the functionality for playing sound is being used, is in the **Tale** (Speech) 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 asociated 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.

4.2.6 Further Reading:

4.3 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 Tab Listener. In this project we did so in a separate class file, using an existing Tab Listener from (Insert Book Reference here). Using the Tab listener you can instantiate an Action Bar as you can see in the following code example.

```
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);
```

In order of lines of code:

1. Get the activities Action bar through reference.
2. Establishing that the action bar is to be used for Tabs by setting its Navigation Mode.
3. Something.
4. Creating a New Tab.
5. Setting the Text that should be seen on the Tab.
6. Constructing a Tab listener so that when you click on this Tab or others, the system know what to do.
7. Finally, Add the Tab to the Action Bar.

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 an User has the right to see a Tab. The Order of the Tabs should therefore match up with the order in the Rights Array.

Further Reading:

4.4 Colour Picker

4.4.1 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 colours of different parts of the application.

4.4.2 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 they use an Android library project called AmbilWarna in their own project, which has been a great help to implementing it ourselves. The AmbilWarna library provides the application with a dialog box that allows the user to pick a colour as an input to the system.

4.4.3 Execution

The use of the AmbilWarna dialog is best seen in the optionsfragment, more precisely when the **Change Category Colour** 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      });
```

As seen in the code, if the button is clicked, a new AmbilWarna dialog will pop up. The default selected colour of the dialog will be

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

, which 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 colour that the user picked in the dialog.

4.4.4 Result

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

4.4.5 Notes:

Currently, the AmbilWarna dialog is used in the Options fragment to change the colour of some of the items in the Speechboard fragment. It is also used in the Management fragment to change the colour of the categories asociated with a user.
Note that changing the colour 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.

4.4.6 Further Reading:

4.5 Displaying Pictograms

4.5.1 Problem

We need a way 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 peices so that one can easily tell what it represents. Areas where we will be displaying pictograms are the Talk board and the Management board.

4.5.2 Solution

Using GridView we can display both the pictograms and its text. Rather than having the GridView display pictograms, 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.

4.5.3 Execution

In order to determine what to show in a GridView, an adapter is needed. Therefore we constructed the Pictogram Adapter in our project, by extending the “BaseAdapter” class. In code segment .... you can see the getView Method, who’s function 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   }
```

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" who's job it is to prepare a view for the application. In this case the pictogramview found through id reference and 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 code fragment ... is 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    }
```

4.5.4 Result

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

4.5.5 Notes

The input view “convertView” in the getView method is an leftover from an earlier iteration of the method. it no longer serves a purpose but “BaseAdapter” requires that the function exists with its current inputs.

4.5.6 Further Reading

4.6

Sentence Board.

4.6.1 Problem:

For kids 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 have 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.

4.6.2 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"picograms that are there to show that there currently are no pictogram at specific spaces.

4.6.3 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 it self. These listeners are seen I codesnippet REFFER TO CODESNIPPET HERE.
Here we also draw the shadow so that the user see that they are dragging a pictogram. The rest of the functionality are in the BoxDragListener. REFFER TO DRAG AND DROP
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 are dropped. We do this by getting the position of the drop, and then what index the object at this position in the sentence board have. This can be seen in codesnippet REFFER TO CODESNIPPET.
We then check if there already is a pictogram in the given place. We can do this by checking if the current pictogram are the "empty"pictogram. If it is not we know there already are a pictogram and we replace it. This is seen in codesnippet REFFER TO CODESNIPPET.
If it is the "empty"pictogram it will try to place the pictogram furthes 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 are placed here and the loop breaks. This is shown in codesnippet REFFER TO CODESNIPPET.
When we know that a pictogram are from the sentence board and are 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 are beign dragged from the sentence board it is removed from it. It can the 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 shift all pictograms that are to the right one to the left so that no spaces are left in the sentence. The deletion of a pictogram are seen in codesnippet REFFER TO FIRST CODESNIPPET.

4.6.4 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.

4.6.5 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 have we 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.

4.6.6 Further Reading:

These functionalities are closely bound to the Drag And Drop functionality REFFER TO DRAG AND DROP.

4.7 Data management

4.7.1 Problem

In PARROT, we uses a lot of classes specifically desgined for the given situation, such as the Pictogram class, the Category class, and the PARROTProfile class. These classes contain the exact ammount 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.

4.7.2 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 nescesary to transfer informataion between PARROT and the database. For instance, whenever the application is started, all information about the current user is loaded from the database.

4.7.3 Execution

In order to demonstrate the functionality of PARROTDataLoader, a few pieces of the code will be shown. However, since the class is fairly long, not all will be shown. 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**.

```
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,col,iconString));
41                  number++;
42              }
43
44              return parrotUser;
```

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

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 firstname 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 REF TIL ADMIN, but suffice to say that it is a wrapper class for a hash table which we use to store data. The Setting object specialSettings retuned 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 colour, 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 desrcibe **loadCategory**.

```
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    }
```

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 contruct a new Category object cat from the name and colour 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**. To explain what it is doing, here is 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.

```
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     }
```

The job 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 asociation 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 initialise 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.

4.7.4 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.

4.7.5 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 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.

4.7.6 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 a working state running, it is possible to make the tests of the Source Code.

BILAG 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.

BILAG B

INVITATION TO USABILITY TEST

Invitation til deltagelse i test af GIRAF

- En Android applikation bygget til børn med autisme

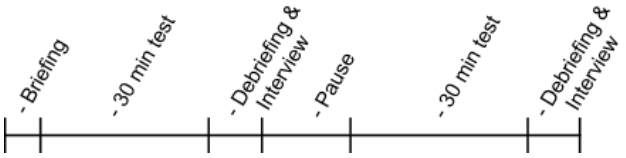
Kære ekspert

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

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

LITTERATUR

[1] Scrum Alliance. Advice on conducting the scrum of scrums meeting, 2011.

[2] Scrum Alliance. Scrum alliance, 2011.

[3] Temple Grandin. Teaching tips for children and adults with autism, December 2002.

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

[5] Samsung. Samsung tablet.

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

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