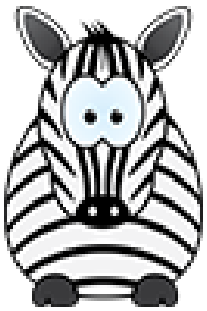
Sekvens

&

Sequenceviewer

Making and displaying sequences of pictograms



Software 6

Casper Holst Laustsen

Johan Leth Gregersen

Morten Møller Jakobsen

May 28th, 2014

This page is intentionally left blank

Aalborg University
The Faculty of Engineering and Science
School of Information and Communication Technology (SICT)
Selma Lagerlöfsvej 300
9220, Aalborg Ø
Phone number: 99407228
http://www.sict.aau.dk

Title: Sekvens & Sequenceviewer
Subject: Making and displaying sequences of pictograms
Project period: February 3rd, 2014 - May 28th, 2014
Project group: sw607f14
Writers:

Johan L. Gregersen

Morten M. Jakobsen

Casper H. Laustsen

Advisor:
Jinling Jiang

Copies: 2
Pages: 95
Appendices: 7
Ended on: May 28th, 2014

Abstract:

For the last few years, Software 6 students have worked on a series of applications under the common name GIRAF. These applications are designed to help people suffering from autistic spectrum disorder and their guardians throughout their day.
This report covers the development of the application Sekvens, designed to help create series of activities through sequences of pictograms. The program is taken from a state where it effectively was a standalone application, to a near final product, tied into the GIRAF solution.
This includes setting it up with GIRAF and a common database. Furthermore, there has been development of additional features based on customer requests. These includes new features to sequences such as the ability to make a choice, and management features such as being able to copy a sequence from one user to another.

This page is intentionally left blank

Reading guide

To understand this report in its entirety, some basic programming knowledge is required, although most parts can be read without any special prerequisites.

The report is built based on the product backlog following the scrum model used (See section 2.1). This means that the report for the most part is split into sprints and issues that have been worked on. The full product backlog can be seen in Appendix A.

When reading this report, it is important to note the described development is only one part of a multi-project split across several groups. Some groups are mentioned in the report by functionality rather than name as it is easier to read. These groups are as follows:

sw601f14: Requirements Responsible for customer contact and obtaining requirements for all groups.

sw602f14: Livshistorier Developers for the Livshistorier application, used to create stories from pictograms.

sw603f14: GComponents Responsible for creating graphical components that all groups can access and use.

sw605f14: GIRAF Developers for the launcher front-end application which serves as a home screen for the applications within the GIRAF multi-project.

sw607f14: Sekvens and Sequenceviewer (Zebra) Our own group, responsible for the Sekvens application and later in the project also Sequenceviewer. Sekvens was called Zebra under previous semesters, and the name can occur throughout the report, especially when dealing with issues from previous semesters.

sw609f14: Kategoriværktøjet Developers of a tool where users can sort pictograms into categories.

sw610f14: iOS Developers for an application similar to Sekvens, but for the iOS platform.

sw611f14: PictoSearch Developers for a search program, designed to find pictograms in the database.

sw614f14: OasisLib Responsible for the local database and the models associated with it.

sw615f14: PiktoOplæser Developers of a communication tool designed to read sequences of pictograms for the user.

Whenever "we" are mentioned in the report, this refers to the Sekvens group, sw607f14.

Because Sekvens originally had certain name conventions throughout the codebase, these has been preserved in the report. These are the relevant terms to know:

- Guardian: An adult who is a caretaker for a person suffering from autistic spectrum disorder. This can both represent parents and pedagogues.
- Child: A person suffering from autistic spectrum disorder.

This page is intentionally left blank

Table of Contents

Chapter 1

Introduction

The proposed project is a multi-project involving all students of the 6th Software semester. The overall purpose of the project, called GIRAF, is to create a set of applications for the Android platform, designed to help people suffering from autism spectrum disorder and their guardians throughout their day. Examples of this are applications like Stemmespillet, which helps people control voice level, and PiktoOplæser which helps them communicate through pictograms and audio. Previous Software 6 students have also worked on the GIRAF multi-project, the logo of which can be seen in figure 1.1. This means that from the beginning of the semester there was an existing codebase to work on. While previous students have developed several applications, none of them was in a state where they were ready for release. In general, most applications had crash issues and were missing a required connection to the designed database. Some applications were missing features or had implemented some bad workarounds to make them run. It was decided that the focus of this semester should be to refactor and rework these programs to a degree where they could be released. Therefore, no new applications was made this semester. The development has been based on requests and feedback from customers. Customers consist of guardians working with autistic people, primarily children. This means that at the end of the semester, the applications ideally consist of tools that both the guardians and children can use effectively.

Figure 1.1: The GIRAF multi-project icon.

To give an example of the applications and their interaction with each other in the GIRAF multi-project, figure 1.2 displays how Sekvens depends on Launcher, GUI, and OasisLib, and utilizes Sequenceviewer, and PictoSearch.

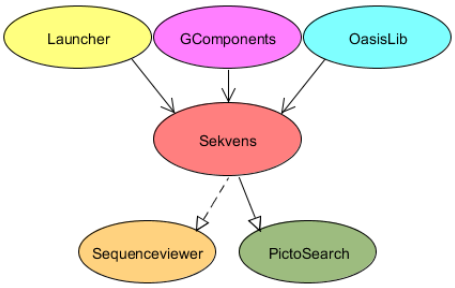


Figure 1.2: The position of Sekvens in the multi-project after Sprint 4.

Chapter 2

Sprint 1

Sekvens is an application that allows guardians to present sequences of pictograms to autistic children. This aids the autistic children in performing daily routines, which they would not be capable of without a visualization. Examples of sequences could be a sequence for how to wash hands, how to put on outdoor clothing during winter, or how to behave during dinner. Sekvens is a digital solution to an existing analog solution which is currently used in institutions in and near the city of Aalborg.

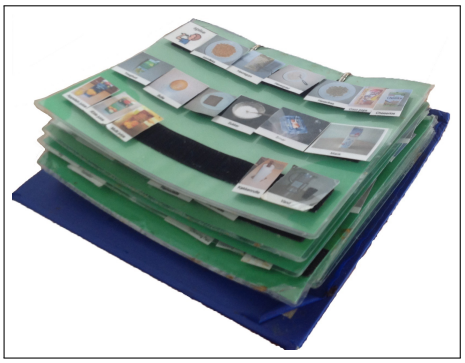


Figure 2.1: A pictogram binder with already constructed sequences

Figure 2.2: Pictogram containers and personal sequences for the children

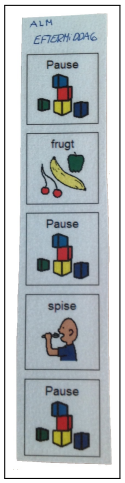


Figure 2.3: A sequence of pictograms made for eating an afternoon meal

It is quite obvious, that the current solution is exhaustive (An example of a current solution is portrayed in figure 2.1,2.2 and 2.3). It was previously decided by a group of software students on 6th semester, to work on a digital solution. They created a first draft of the application that we picked up. They had a standalone application running, but had remaining issues and obstacles they did not have time to solve.

During sprint 1, our goal was to solve the remaining issues and obstacles for the application rather than developing new features or creating additional functionality for already existing features. This was a common agreement among the software 6 students working on the GIRAF multi-project.

2.1 Project Management

To manage the GIRAF multi-project it was chosen to use a variant of Scrum. The version divided the semester into sprints with different focuses on each sprint (such as refactoring in sprint 1). At the end of each sprint, each group would present their work and issues to the customers and other groups. Furthermore we would start planning the next sprint. We used an issue-tracker called Redmine to manage the projects within the GIRAF multi-project, along with a Jenkins builder to monitor compile errors.
We also decided to use Scrum to manage the development of Sekvens. This meant that we held daily stand-up meetings to discuss what everyone was working on, how it was going and whether they needed help from the group to solve their issues. Scrum uses a product backlog to track issues for the product. Scrum also uses a sprint backlog, which contains the issues planned for the current sprint. We also used an internal group issue-tracker called AgileWrap, to keep track of the issues in the multi-project but also additional issues regarding research and so on. The issues from AgileWrap are not listed in this report [1].

2.2 Project Planning

We took part in weekly status-meetings during the entire multi-project, which had the purpose of keeping the multi-project on track and make general decisions regarding GIRAF. During these status-meetings the length of each sprint was decided. We used the sprint length to calculate the amount of hours available to burn, and saved it in our issue tracker. At the status-meetings the groups also agreed on a focus for each sprint.

Before sprint 1 we ran over the issues in the existing issue tracker and decided on which to include in the sprint backlog for the first sprint. Furthermore we reviewed the code of Sekvens, and added issues as we stumbled upon them. The decision on what to include, was based on the hours estimated, the priority for the issue, and the focus for the sprint. We favored issues with urgent or high priority for sprint 1. We chose roughly as many estimated hours of issues as there were hours to burn.

2.3 Design Patterns

Design patterns also known as best practices is "a general repeatable solution to a commonly occurring problem in software design" [2]. Having good design patterns can speed up development and in general help readability of the code. Although there was no obvious patterns in the initial code of Sekvens, it was decided to attempt following a few patterns anyway. Having a messy codebase does not justify creating more mess. At the beginning of the semester, it was a challenge to decide on design patterns because it was not known what the development would consist of. This meant the design patterns had to be general and not application specific. The agreed upon design patterns are listed as follows:

Object Pool Whenever possible, objects should be re-used rather than creating new instances. This can save some potentially expensive operations.

Singleton Activities should be always be singleton (Single instance), as it could be confusing for the customers to have more than one instance of these classes.

Private class data Classes should generally be hidden and not generally accessible directly. This means that under normal circumstances, variables and methods should be private. While some methods often are needed to be public, class variables should always be private and only manipulated using a get/set method.

Observer Since Sekvens consisted of lists of sequences and children we want to make sure that there is a way to update these automatically.

State Sekvens was entirely split up into two major states: Child and Guardian mode. Allowing certain classes to change behavior depending on state was a must.

Template method Whenever something can be broken into a meaningful and readable sub-component, this should usually be done. This helps give access to code re-usability and easier management of components.

2.4 Sprint Overview

The following section is a description of work made in sprint 1.

2.4.1 Sprint Backlog

After a discussion and evaluation of project backlog, we agreed on the sprint backlog for the first sprint. The biggest issue in sprint 1 is analyzing and fixing the TODO's in the already existing code -furthermore getting an overview of how the code is structured and how the program is built.

ID	Issues	Es. hours
2	Updating a picture in sequence -> no update in overview	8 hours
4	Destroy Sekvens if minimized	8 hours
6	Changing application name	1 hour
10	Changing a sequence name is not intuitive	8 hours
11	Analyzing TODOs in code	16 hours
14	Import temporary pictures into Sequence	4 hours
17	Solve TODO in SequenceViewGroup - Can child be bigger than parent?	2 hours
18	Child selected upon opening the program is not highlighted	2 hours
19	Zebra is installed with two app-icons	16 hours
20	When a sequences is created it should have a return to overview button	8 hours

Table 2.1: This is a list of the issues we included in our sprint backlog during sprint 1

2.4.2 Resolved Issues

ID	Issue	Es. hours
6	Changing application name	1 hour

Table 2.2: Issue ID 6

This was a trivial task of changing the application name in the AndroidManifest of the project. The project is as mentioned now called Sekvens as opposed to the former name Zebra. The change was made due to one of the very first customer requests; the applications within GIRAF should have eloquent names, rather than names from the earlier savannah animal theme.

ID	Issue	Es. hours
10	Changing a sequence name is not intuitive	8 hours

Table 2.3: Issue ID 10

Changing the name of a sequence was previously done by clicking on the title text, but it was not obvious that it was in fact editable. The issue was resolved by adding a button (as shown in figure 2.4) which programmatic places the cursor in the textfield and popping up the virtual keyboard. We felt like this cluttered the entire GUI with too many buttons, so we decided to keep the issue open for later revision. No icons displays editing the name therefore we used a simple white placeholder with the word 'TEXT' on it.

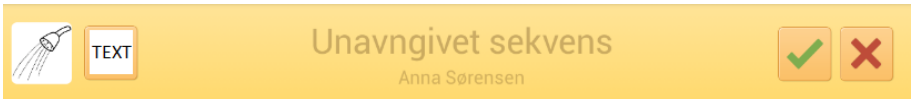


Figure 2.4: This is a picture of top bar after the edit button was added

ID	Issue	Es. hours
11	Analyzing TODOs in code	16 hours
17	Solve TODO in SequenceViewGroup - Can child be bigger than parent?	2 hours

Table 2.4: Issue ID 11 and 17

Solving the TODO's of the previous group who worked on the project was one of the more difficult issues. They were usually very unclear, examples being `//TODO: ???` and `//TODO: this is ridiculous`. Since statements were this vague and seemingly not meant for other groups to understand, these issues were time consuming. The estimate of 16 hours however fit surprisingly well for both analyzing and solving the issues. This is partly due to some of the previous groups lack of understanding for Android programming. These TODO's could thus just be removed. Issue 17 was created as a separate task because it was difficult to troubleshoot, but it was found to that code already existed to ensure that a child view could not be bigger than the parent view.

ID	Issue	Es. hours
14	Import temporary pictures into Sequence	4 hours

Table 2.5: Issue ID 14

While Sekvens was runnable from the beginning of the sprint, it had no pictograms available. This did not change during sprint 1. Therefore, having some local images to use in the application would be beneficial. It was discovered that one of the previous groups had hardcoded paths and names for pictograms that we had no access to. Pasting new images with matching titles on the hardcoded path allowed us to have images to work with in the application.

ID	Issue	Es. hours
18	Child selected upon opening the program is not highlighted	2 hours

Table 2.6: Issue ID 18

This issue was looked into but turned out to be difficult to solve. However, while it was in fact never fixed, it also became a non-issue due to new requirements. These requirements specified that

the child list should not be available from within Sekvens. Choosing children should be done from the Launcher application. The issue was therefore closed.

ID	Issues	Es. hours
19	Zebra is installed with two app-icons	16 hours

Table 2.7: Issue ID 19

Debugging why the application was installed twice upon compiling was another hard issue. This was one of the first issues we attempted to solve, and lack of knowledge about Android made this simple fix a time-wise underestimated issue. We did not immediately realize that the issue was not within Sekvens, but in one of its dependencies. Once discovering the cause it was a simple fix of removing one line from a submodules Android Manifest.

ID	Issue	Es. hours
20	When a sequences is created it should have a return to overview button	8 hours

Table 2.8: Issue ID 20

When displaying a created sequence it was not apparent how the user was able to return to the overview. The former way was by pressing the tablets back button, but it seemed rather unintuitive as it was supposed to go to the overview, and not back to editing the sequence. This was solved by adding a button to allow the user to return and furthermore overriding the tablets button to return to overview (A picture of the button is displayed in figure 2.5).

Figure 2.5: A picture of a simple button in 'Sekvens'

2.4.3 Remaining Issues

The following is the remaining issues which were not completed within the time frame for sprint 1.

ID	Issues	Es. hours
2	Updating a picture in sequence ->; no update in overview	8 hours

Table 2.9: Issue ID 2

This was an issue that we did not look into because of time limitations. The reason this particular issue has been pushed back in favor of others was both because it was not a high priority issue and because having pictograms in Sekvens was a prerequisite, refer issue 14 in table 2.5.

ID	Issues	Es. hours
4	Destroy Sekvens if minimized	8 hours

Table 2.10: Issue ID 4

According to the previous report on Sekvens[3], the home button caused an issue. After using the home button, the guardian found it confusing that when reopening Sekvens it resumed at the previous state of the application. It was preferred that the application should reopen as if it was just launched. This issue was looked into and an attempt to solve it was made by destroying the

application when the home button was pressed. This however caused a new bug where Sekvens would be destroyed after accessing PictoSearch from the application. The proposed solution was thus discarded.

2.5 Sprint Evaluation

Sprint 1 was influenced by the startup of the multi-project. Time was spent organizing groups, distributing project and researching previous work. The result of this research showed a running application for creating sequences, but with limited functionality and missing connections with other multi-project projects like the Launcher and OasisLib. We collected issues for a product backlog which is displayed in table A.1. We also gathered requirements to Sekvens from the previous semester. We also investigated further requirements to the application by compiling a series of questions and choices for the guardians to choose among, which we handed to the requirements-group to resolve (see Appendix B).

We resolved a list of issues during sprint 1, which can be seen in subsection 2.4.2. How these issues were resolved are documented below each issue. The list of issues, which were not resolved are described in subsection 2.4.3, and describes why we did not resolve the issue.

Chapter 3

Sprint 2

In sprint 2 we focused mainly on improving the design of Sequence as well as plan for the future. This resulted in arranging meetings with other groups to discuss the user interface design of our applications. These are the groups that have a similar interface to Sekvens. We arranged meetings with Livshistorier, and the iOS group. We have described the collaborations in chapter 6. Another main task in this sprint was to design the database for storing sequences. This was also discussed with the Lifestories, PiktoOplæser, and the iOS group. This is described in section 6.1.

3.1 Sprint Overview

The following section is a description of work made in sprint 2.

3.1.1 Sprint Backlog

The following is a list of the issues we have chosen work with during sprint 2. The sprint focuses mainly on GUI changes and cross group communication.

ID	Issues	Es. hours
1	Syncronization with the Database	32 hours
15	Original drag location flickers when rearranging	32 hours
21	View-mode created	128 hours
25	Fix ADD/SAVE/BACK button in SequenceActivity	4 hours
26	Change all buttons	8 hours
27	Change GUI of Viewgroups	16 hours
28	Remove the Childlist	32 hours
29	Update to Oasis	4 hours
30	Refactor overview	32 hours
31	Rework sequenceTitle box to look editable	4 hours
32	Display some temporary sequences in overview	4 hours
33	Nested Sequences	64 hours
34	Choices in Sequnces	16 hours
35	After adding pictograms to a sequence it should not go to the ”sequenceActivity”, it should go to ”MainActivity”	8 hours

Table 3.1: This is a list of the issues we included in our sprint backlog during sprint 2

3.1.2 Resolved Issues

ID	Issues	Es. hours
15	Original drag location flickers when rearranging	32 hours

Table 3.2: Issue ID 15

This issue was reported by the group working on Sekvens during the previous semester. At this point we had been working with Sekvens for two sprints without experiencing this issue. We looked into it and analyzed it and concluded the problem was resolved during the previous semester. The issue was resolved but never logged on Redmine.

ID	Issues	Es. hours
26	Change all buttons	8 hours
27	Change GUI of Viewgroups	16 hours

Table 3.3: Issues ID 26 and 27

One of the biggest changes from sprint 1 is for the GUI in Sekvens to match the overall theme of GIRAF. This meant changing every single button and grid to the templates GComponents provided. There was a lot of GUI to change, and because of this we chose to just describe three examples. The first example of one of the completely customizable dialog boxes is shown in listing 3.1 and displayed in figure 3.1. The example shown is when the guardian tries to exit editing a sequence without having saved. Each button is linked to an XML file determining where the button is displayed in the layout as well as what the headline and description says.

```
1
2  public class backDialog extends GDialog {
3
4      public backDialog(Context context) {
5
6          super(context);
7
8          this.SetView(LayoutInflater.from(this.getContext()).inflate(R.layout
               .exit_sequence_dialog,null));
9
10         GButton saveChanges = (GButton) findViewById(R.id.save_changes);
11         GButton discardChanges = (GButton) findViewById(R.id.discard_changes
               );
12         GButton returntoEditting = (GButton) findViewById(R.id.
               return_to_editing);
13
14
15         saveChanges.setOnClickListener(new GButton.OnClickListener() {
16
17             @Override
18             public void onClick(View v) {
19                 SequenceActivity.this.saveChanges();
20             }
21         });
22
23         discardChanges.setOnClickListener(new GButton.OnClickListener() {
24
25             @Override
```

```
26              public void onClick(View v) {
27                  finish();
28              }
29          });
30
31          returntoEditting.setOnClickListener(new GButton.OnClickListener() {
32
33              @Override
34              public void onClick(View v) {
35                  dismiss();
36              }
37          });
38      }
39  }
```

Listing 3.1: Customizable Dialogbox

Figure 3.1: This dialog box is the result of the code in the listing above

The GComponents group also provided a list of simple templates e.g. a 2 button `GDialogMessage`. The example shown in 3.2 is the delete button shown in the overview. The constructor needs the context, an image, a title, a description and a functionality for one of the buttons. The second button added in the dialog box is a button that closes the dialog and returns to the previous state.

```
1
2  public void deleteSequenceDialog (View v) {
3      GDialogMessage deleteSequence = new GDialogMessage(v.getContext(),
4      R.drawable.ic_launcher,
5      "Slet Sekvens",
6      "Du er ved at slette sekvensen, er du sikker paa du vil det?",
7      new View.OnClickListener() {
8          @Override
9          public void onClick(View v) {
10             deleteSequence();
11         }
12     });
13 }
```

Listing 3.2: GDialogmessage

The GComponents group also provided an easy implementation of the buttons. They provided an extension, `GButton` of the Android `Button`. The implementation was simply refactoring every button to be an instance of GButton instead of Button. A couple of examples are shown in figure 3.2. In the same vein they provided `GGridView` as an extension to the Android `GridView`. An example of this is show in 3.3.

Figure 3.2: This is an example of 3 new buttons without final icons

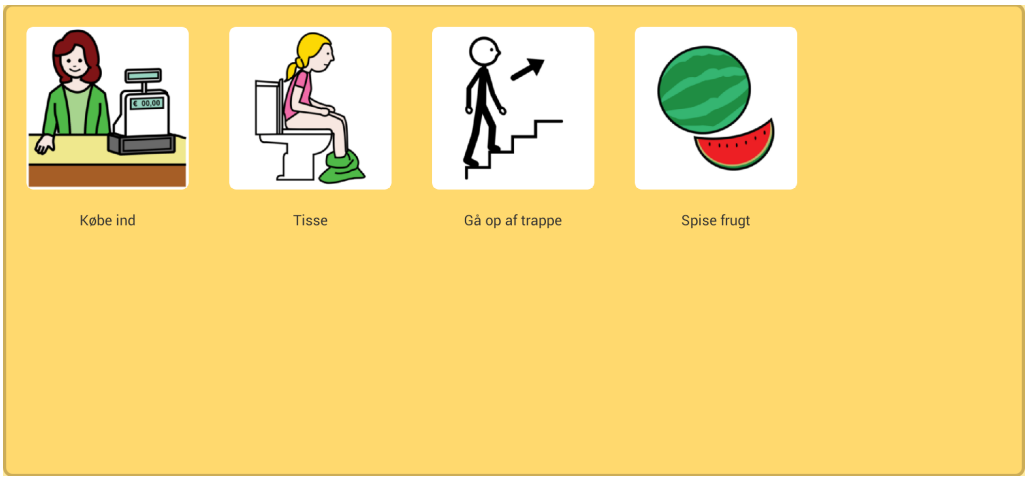


Figure 3.3: This is the grid in MainActivity of a child with 4 sequences attached

ID	Issues	Es. hours
28	Remove the Childlist	32 hours

Table 3.4: Issue ID 28

It was originally possible to select a child within the application through a menu on the left side of the screen. This was never the intention since the Launcher application handles both the guardian login and which child to work with. As a result, the entire menu has been removed and Sekvens now takes a child as `intent` from the Launcher instead. Figure 3.4 and figure 3.5 is a side by side comparison of the overview with and without the childlist.

Figure 3.4: A picture of the overview with a ChildList

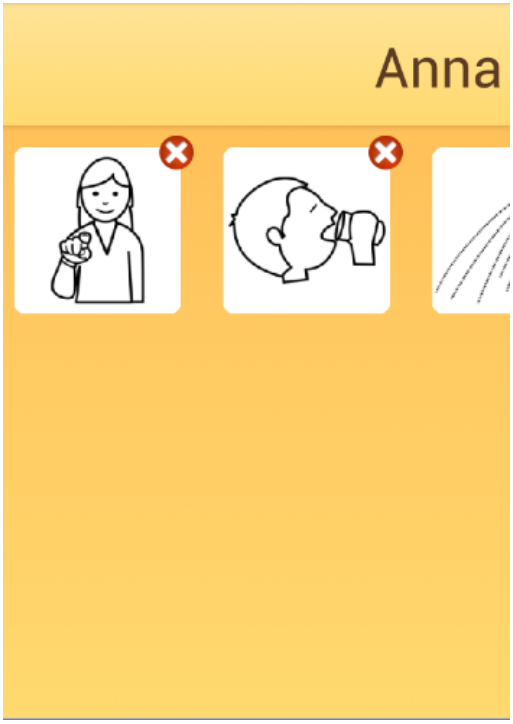


Figure 3.5: A picture of the overview without a ChildList

ID	Issues	Es. hours
29	Update to OasisLib	4 hours

Table 3.5: Issue ID 29

OasisLib received major updates in the beginning of sprint 2. This meant that our application no longer compiled as certain dependencies was changed. Modifying Sekvens to match the OasisLib update mostly revolved around changed data-types, which had to be updated throughout the entire application. The database did as mentioned not support saving pictograms to sequences, so every sequence is saved as multiple pictograms with their respective positions in the sequence.

ID	Issues	Es. hours
30	Refactor overview	32 hours

Table 3.6: Issue ID 30

In continuation of removing the child list (see table 3.4) Sekvens needed to be setup to take the ID of a child and a guardian from the Launcher. As a child could no longer be chosen from within Sekvens, the application would have to be opened from Launcher from this point on. The functionality was implemented by receiving the intent sent from the launcher, and parsing the extras into two locally saved ID's `-currentGuardianID` and a `currentChildID`. The child in the database matching the `currentChildID` would be the profile that Sekvens should load and manage sequences for.

```
1
2   private void setChild() {
3     sequences.clear();
4     Bundle extras = getIntent().getExtras();
5         guardianId = extras.getInt("currentGuardianID");
6
7         //TODO: childId from launcher is currently long, but we expect it to
                be int soon. This is why we parse it here.
8         long childIdLong = extras.getLong("currentChildID");
9         int childId = Integer.parseInt(Long.toString(childIdLong));
10
11        try{helper = new Helper(this);}
12        catch(Exception e){}
13      Profile guardian = helper.profilesHelper.getProfileById(guardianId);
14      List<Profile> childProfiles = helper.profilesHelper.
            getChildrenByGuardian(guardian);
15
16      for (Profile p : childProfiles) {
17        if (p.getId() == childId) {
18                Child c = new Child(childId, p.getName(), p.getImage());
19                String name = p.getName();
20                Bitmap picture = p.getImage();
21                selectedChild = c;
22            }
23       }
24        refreshSelectedChild();
25   }
```

Listing 3.3: A small code example of fetching the intents from Launcher

ID	Issues	Es. hours
31	Rework sequenceTitle box to look editable	4 hours

Table 3.7: Issue ID 31

Changing the name of a sequence was originally not intuitive. It was not visible that the text field was editable, and by request of guardians this was changed. To make it intuitive we first made a button which would put the cursor in the text field and pop up the keyboard, but found that it was unnecessary cluttering of the screen. Later in the sprint this was reworked. Instead, the text field was given a plain white background and a frame similar to buttons, making it stand out as something that can be changed.

ID	Issues	Es. hours
32	Display some temporary sequences in overview	4 hours

Table 3.8: Issue ID 32

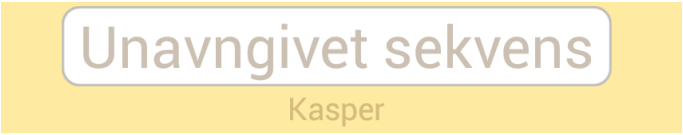


Figure 3.6: This is a picture of the text field in which it is possible to edit the name of a sequence

This issue was made because OasisLib were not handling sequences. Not having any sequences in the application caused a problem where we were not able to access some parts of the application at runtime. Since that the application has its own model of sequences, it was possible to resolve this by creating temporary sequences to display while working with the application. The code for creating fake sequences is shown in 3.4

```
1
2  private void loadSequences() {
3      //TODO createFakeSequences is a temporary fix to generate some Sequences
4      List<Sequence> list; // = SequenceFileStore.getSequences(this,
           selectedChild);
5      list = createFakeSequences();
6      selectedChild.setSequences(list);
7  }
8
9  private List<Sequence> createFakeSequences() {
10
11     Sequence s = new Sequence();
12     s.setTitle("TEST SEQUENCE");
13     s.setImageId(10);
14     s.setSequenceId(5);
15
16     Pictogram a = new Pictogram();
17     Pictogram b = new Pictogram();
18     Pictogram c = new Pictogram();
19     a.setPictogramId(0);
20     b.setPictogramId(1);
21     c.setPictogramId(2);
22     s.addPictogramAtEnd(a);
23     s.addPictogramAtEnd(b);
24     s.addPictogramAtEnd(c);
25
26     List <Sequence> list = sequences;
27     for (int i = 0; i < 12; i++) {
28         list.add(s);
29     }
30     return list;
31 }
```

Listing 3.4: The code in the example is the code used to create fake sequences

ID	Issues	Es. hours
34	Choices in Sequences	16 hours

Table 3.9: Issue ID 34

The more independent users can easily handle sequences covering a larger timespan. These sort of sequences require a possibility to create nested sequences e.g. a morning schema could entail smaller sequences such as eat breakfast, or take a shower. When the autistic people have a period of free time, they are sometimes presented with a choice (e.g. playing with cars or drawing). This issue has been solved by allowing the guardian to add already created sequences, as well as choices, instead of only being able to add pictograms. If a sequence is chosen, the user is redirected to a new MainActivity (without all of the buttons), and you can choose from existing sequences. Once a sequence is chosen, it is a nested sequence, i.e. a sequence within another sequence.

If the user chooses to add a choice instead of a sequence or pictogram, he will be presented with another dialog box. This dialog box contained a SequenceViewGroup similar to the one in SequenceActivity, where the user is able to add pictograms to the choice. Once the choice is created, the child viewing the finished sequence should be able to make a choice between the added pictograms.

As of the end of sprint 2 the database does not fully support sequences which means the full functionality could not be added until sprint 3. This means the issue regarding nested sequences could not be closed completely yet.

Figure 3.7: This is the dialog box making it possible to add a choice or a sequence into another sequence

ID	Issues	Es. hours
35	After adding pictograms to a sequence it should not go to the "sequenceActivity", it should go to "MainActivity"	8 hours

Table 3.10: Issue ID 35

This issue was opened upon request by the requirement group who talked to the guardians. They stated that it was not intuitive to launch the sequence in the sequenceActivity. It was changed to return to the overview, which displays a grid of all sequences. This was done by inserting a finish of the sequenceActivity.

3.1.3 Remaining Issues

Here are the remaining issues that we did not make in this sprint.

ID	Issues	Es. hours
1	Syncronization with the Database	32 hours

Table 3.11: Issue 1

Syncronization with the database is still in process because the database was still under development, as discussed in 6.1. We put this in the sprint because we thought sequences were already represented in the database.

ID	Issues	Es. hours
21	View-mode created	128 hours

Table 3.12: Issue 21

The issue was created midway through sprint 2, and were under discussion for a while. The entire collaboration with Livshistorier, PiktoOplæser and the iOS group is described in section 6.2.

ID	Issues	Es. hours
25	Fix ADD/SAVE/BACK button in SequenceActivity	4 hours

Table 3.13: Issue 25

This issue was still iced as the database synchronization was also iced.

ID	Issues	Es. hours
33	Nested Sequences	64 hours

Table 3.14: Issue 33

Nested sequences was almost done, but because we could not save sequences into the database yet, we could not resolve this issue.

3.2 Sprint Evaluation

In this sprint we collaborated with the Livshistorier-, the iOS-, and the PiktoOplæser groups in order to structure a new database model (see section 6.1) and negotiate the terms of the new project 'Sequenceviewer' (see section 6.2). We were assigned as the group to work on Sequenceviewer, which would be a general template for showing sequences. The OasisLib group did make some minor changes to their functionality which meant we had to adapt Sekvens to it. This meant we had to spend time adjusting (see section 6.4).

We made many cosmetic changes and added a few new functionalities. These can be seen in section 3.1.2 about resolved issues. In the sprint-end presentation the stakeholder questioned about the functionality for making a choice between pictograms and making nested sequences. Therefore we had to specify what we actually meant with these functions. Another important thing was to make sure that the we did not show any half pictograms -meaning we have to create some sort of snapping to make sure that half a pictogram is never shown.

All groups received their official requirements and translated them into issues for this sprint, which can be seen in appendix D. We also wrote a report to the requirement group, because we wanted some specific feedback. The questions can be seen in appendix B, and the answer from the requirements group can be seen in appendix C.

Chapter 4

Sprint 3

For sprint 3 the focus of development changed to focus more on Sequenceviewer, as this would be a dependency for other groups as well as ours. As this had only just been decided in sprint 2, it was a project that had to be started up from scratch. For Sequenceviewer, sprint 3 thus revolved around setting up the core of the activity. For Sekvens, the focus mostly revolved about polishing the application, preparing it as well as possible for the upcoming database support.

4.1 Sprint Overview

The following section is a description of work done in sprint 3.

4.1.1 Sprint Backlog

ID	Issues	Es. hours
3	Cancel and replace action function	12 hours
4	Destroy Sekvens if minimized	8 hours.
12	Make add pictogram button placement intuitive	16 hours
21	Create view-mode	128 hours
34	Implement choices in sequences	16 hours
36	Create snap in SequenceActivity	4 hours
37	Create snap in SequenceGrid	4 hours
38	Create nested sequence placeholder	2 hours
39	Create choice placeholder	2 hours
40	Create exit button for Sekvens	1 hours
43	Remove sequence name button	0.5 hours
47	Sequenceviewer: Dynamic show	16 hours
48	Sequenceviewer: possible settings for up to 7 pictograms	16 hours
49	Sequenceviewer: highlighting with a outline.	16 hours
50	Sequenceviewer: an arrow highlighting method	16 hours
51	Sequenceviewer: in settings dialog - choose what type of highlighting there should be	2 hours
52	Sequenceviewer: show only whole pictograms	8 hours
53	Sequenceviewer: blur pictures out of focus	8 hours

Table 4.1: This is the list of issues we included in our sprint backlog during sprint 3

4.1.2 Resolved Issues

ID	Issues	Es. hours
3	Cancel and replace action function	12 hours

Table 4.2: Issue ID 3

This issue is part of the clients requirements for the application. We initially gave it a low priority as we thought this requirement was sub-par. If a guardian wants to cancel or replace a pictogram, sequence or choice, they could simply create a new altered sequence or alter the existing one. In the end, the issue was closed as we believe the functionality was achieved through different ways.

This could be a project for future groups if the clients find the requirement fundamental enough to be an issue.

ID	Issues	Es. hours
4	Destroy Sekvens if minimized	8 hours

Table 4.3: Issue ID 4

A customer request was that if the application is ever minimized, whenever the user attempts to launch it again, it should open a fresh instance rather than returning to where the customer last left it. This turned out to be a more difficult task than assumed. An attempt was made in the first sprint to solve the issue (see table 2.10), but the suggested idea was rejected. The suggested solution was to override Android's `onStop` to also call `onDestroy`. This would definitely fix the issue, as whenever the activity was not in focus, it would be destroyed. It however created the unintended effect that once opening another activity (SequenceActivity, PictoSearch), the original activity, now in the background would have been destroyed. This meant there was nothing to return to, once done with the opened activities. The issue was postponed, as it was not seen as a high priority issue.

The request was re-opened in sprint 3. At this point an attempt was made to override the tablet's Home and Back buttons to kill the application if pressed. It was however discovered that the Home button was atomic. It was neither possible to interrupt it or even set a flag to indicate if pressed. Instead a new variable was introduced to all activities of Sekvens, called `assumeMinimize`. This was a boolean, set to be true by default. In the `onStop` method, it would then be checked. If set to true, the application should be killed. This task included fetching the instances of any other open activities, and killing them as well, before killing the instance in which `onStop` was run.

What made this work, was to set the variable to false whenever a new Activity was opened. In this case, when `onStop` was run, the application would not be killed, but `assumeMinimize` would be reset to true. This behavior was copied to all Activities. The `onStop` code to make `assumeMinimize` work for MainActivity can be seen in 4.1. If MainActivity was launced in `nestedMode` where the guardian can insert a sequence into another, this means that both SequenceActivity and another instance of MainActivity is running in the background. They are then both killed before the activity itself, where `onStop` was run, is killed.

```
1 protected void onStop() {
2     if (assumeMinimize) {
3         //If in NestedMode, kill all open Activities. If not Nested, only
              this Activity needs to be killed
4         if (nestedMode) {
5             SequenceActivity.activityToKill.finish();
6             MainActivity.activityToKill.finish();
7         }
```

```
8               finishActivity();
9       } else {
10          //If assumeMinimize was false, reset it to true
11          assumeMinimize = true;
12      }
13      super.onStop();
14  }
```

Listing 4.1: onStop with the check for assumeMinimize

ID	Issues	Es. hours
12	Make add pictogram button placement intuitive	16 hours

Table 4.4: Issue ID 12

When the customers were editing a sequence, they did not find it intuitive that you had to press the large framed plus at the end of the sequence as shown in figure 4.1. Furthermore when you had a sequence of more than 4 pictograms, you would have to scroll in order to find the plus. We decided to add an alternative to the already existing method, and added a simple button. This way the clients have multiple ways to edit the sequences, and could use whichever they see fit. Figure 4.1 shows the `sequenceGridView` in sequenceActivity.

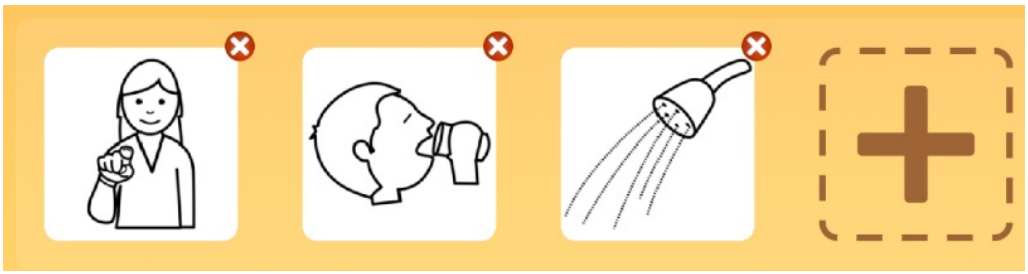


Figure 4.1: A picture of SequenceActivity with the add sequence plus sign

ID	Issues	Es. hours
21	Create view-mode	128 hours

Table 4.5: Issue ID 21

At the beginning of sprint 3 we were aware that some groups were interested in having a shared viewer for their equivalents of sequences. The details were however very vague, and it was not known what the demands would be for such a viewer. Since there was not enough left in Sekvens for two more sprints, we agreed to be responsible for the viewer, which is when this issue was created.

Later in sprint 3 the details were sorted out under a collaboration meeting with the relevant groups. As the different groups needed support for different features, the viewer turned out to be a large component. For this reason, Sequenceviewer was given its own repository and a separate backlog, so that for all intents and purposes we could view it as a project on its own. This means that while this issue was closed, Sequenceviewer was a new project being started up.

ID	Issues	Es. hours
34	Implement choices in sequences	16 hours

Table 4.6: Issue ID 34

As choices were now possible to add in sequences, we needed some way to display and edit choices. Livshistorier had a way to show and display choices, but we found that the theme of their dialog box did not fit well into Sekvens. We wanted to keep editing consistent within the application itself, so we chose to re-use existing `ViewGroups` to make them consistant. We created a fully customizable dialog box and created a new `SequenceViewGroup` within it. The following listing shows how the code is written and portrayed in figure 4.2

```
<HorizontalScrollView
    android:id="@+id/horizontalScrollView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="0dp"
    android:overScrollMode="always"
    android:padding="0dp"
    android:fadeScrollbars="false">

    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent">

        <dk.aau.cs.giraf.zebra.SequenceViewGroup
            android:id="@+id/sequenceViewGroup"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dp"
            android:layout_marginRight="30dp"
            svg:horizontalSpacing="20dp"
            svg:itemHeight="@dimen/activity_main_picto_size"
            svg:itemWidth="@dimen/activity_main_picto_size"
            android:background="@layout/main_picto_container_bg"
            android:layout_gravity="center_vertical">
        </dk.aau.cs.giraf.zebra.SequenceViewGroup>
    </FrameLayout>
</HorizontalScrollView>
```

Re-using the`ViewGroups` made the implementation easy, as all of the functionality needed for choices was already implemented in `SequenceViewGroup`. The features included adding pictograms to the choice, as well as arbitrary scrolling features and handling the views within the `ViewGroup`. The fully customizable dialog box provided by the GComponents group, made it easy to follow the overall theme of GIRAF. Figure 4.2 is a representation of what creating a choice looks like.

ID	Issues	Es. hours
38	Create nested sequence placeholder	2 hours
39	Create choice placeholder	2 hours

Table 4.7: Issue ID 39

With the addition of nested and choice functionality to sequences, we needed a way to easily identify these elements within a sequence. Initially, a frame consisting of a choice would be displayed using the first pictogram of that choice. A frame consisting of a nested sequence would have the first pictogram in that sequence. Without remembering what the frames were, it was impossible to distinguish these special frames from other pictograms in the sequence without clicking on them. This would result in different behavior, depending on what was clicked. To solve this issue, hardcoded placeholders were temporarily created, replacing all pictograms with placeholders.

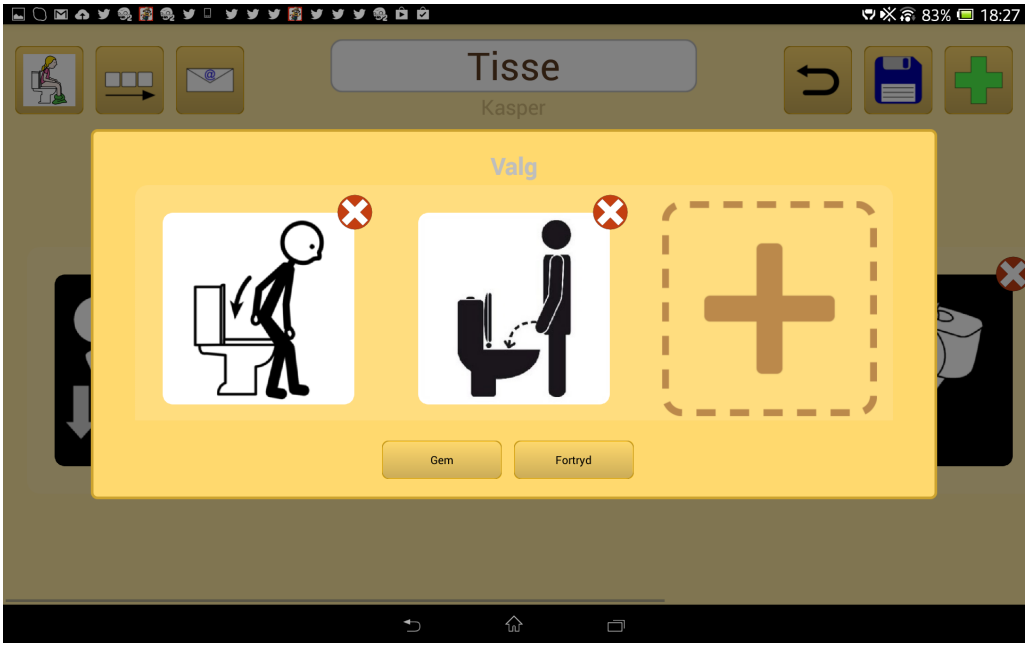


Figure 4.2: This is an example of a choice being created

These would identify them as pictograms, choices or nested sequences. The following figures are the placeholders used in Sekvens.

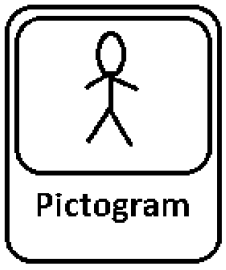


Figure 4.3: The placeholder for a pictogram

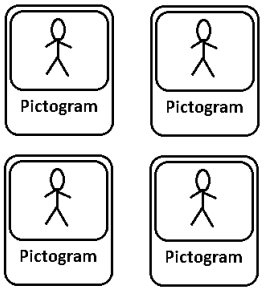


Figure 4.4: The placeholder for a sequence

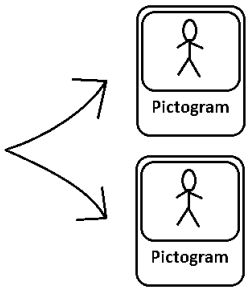


Figure 4.5: The placeholder for a placeholder

ID	Issues	Es. hours
40	Create exit button for Sekvens	1 hours

Table 4.8: Issue ID 40

A request that the customers had for all applications within GIRAF, was that they should have an exit button. It should be noted that this is not intuitive from our point of view. Android handles the lifecycles of applications automatically[4] and the Back button can also perform the desired behavior. The exit button is redundant, but as it is both a customer request and decided upon in the other GIRAF applications, the button was created in Sekvens as well.

ID	Issues	Es. hours
43	Remove sequence name button	0.5 hours

Table 4.9: Issue ID 43

When solving issue 31 described in 3.7, we forgot to remove the original button. This was not discovered before the end of the second sprint, and was not deleted until sprint 3.

ID	Issues	Es. hours
47	Sequenceviewer: Dynamic show	16 hours

Table 4.10: Issue ID 47

Sequenceviewer needed to be able to vary the number of pictograms to display. This issue originated from work done by the Requirements group during the semester, and can be seen in the requirement report in appendix D. The reason for this issue comes from the difference in the abilities that children possess. The different institutions who will work with this system, has children with varying strengths and weaknesses. Some children can only perform one task at a time, and should only have the next activity to look at. Some children however may be capable of seeing the entire sequence of activities at once.

The varying number of pictograms to display, would be based on a setting provided by the application calling Sequenceviewer.

This issue was solved by first using one of two GComponents called GHorizontalScrollViewSnapper or GVerticalScrollViewSnapper. The collaboration with GComponents about creating these two components can be seen in section 6.4. Horizontal is used for making landscape scrolling, and vertical for portrait. As both screen-orientations are built the same way, only landscape-mode is explained here.

When Sequenceviewer is called, it launches `onCreate` together with _main.xml. In `onCreate` a condition checks whether the calling application has requested landscape or portrait, see listing 4.2. In the case of landscape-mode, the layout in landscape_mode.xml is inflated into activity_main.xml. landscape_mode.xml contains the GComponents regarding buttons, images and text for the Sequenceviewer. In `onCreate` the colors from GComponents are set up using the `setColors` method. Then the ScrollView for landscape-mode is created by instantiating a new `HorizontallyScroll` view. `HorizontallyScroll` takes 5 paremeters:

- *Context context*, the current context.
- *Sequence sequence*, the sequence to display. This is gathered from the database by its id, sent from the calling application as `intent`.
- *String type*, the name of the calling application.
- *View p*, the view in which the HorizontallyScroll view is added.

- *int picVisCount*, the number of visible pictograms. Sent by the calling application as `intent`.

The ScrollView is then added to the relative layout in the landscape-layout.

```
1
2  setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
3  View landscape = inflater.inflate(R.layout.landscape_mode, null);
4  mainView.addView(landscape);
5
6  setColors();
7
8  HorizontallyScroll horizontalScrolling = new HorizontallyScroll(this,
       sequence, type, landscape, numberOfVisibiblePictograms);
9  horizontalScrolling.setTag("land_hscrollview");
10 horizontalScrolling.setLayoutParams(new HorizontalScrollView.LayoutParams(
       HorizontalScrollView.LayoutParams.MATCH_PARENT, HorizontalScrollView.
       LayoutParams.MATCH_PARENT));
11 RelativeLayout land_rellay = (RelativeLayout) landscape.findViewById(R.id.
       landscape_rellay);
12 land_rellay.addView(horizontalScrolling);
```

Listing 4.2: Dynamic number of pictograms in landscape-mode

By Android defined standards, multiple views cannot be directly added to a `ScrollView`. Therefore, in the constructor of `HorizontallyScroll`, a `LinearLayout` is created and added to the `ScrollView` called `mainLayout`.

Next thing was to create a view in the `ScrollView` for each pictogram in the sequence. This was done by first overriding `onDraw` in `HorizontallyScroll`. We need to work with the width and height of the view in which the `ScrollView` exists, and Android does not define width and height before it has drawn the view. For each pictogram, we instantiated a new `PictogramView`. Before we added the pictogram to the view, we had to consider the number of pictograms to display. Therefore we scale it. We define the limiting factor in the view by using *math.min*, which returns the most negative of two argument. We feed *math.min* with `getWidth() / numberOfVisibiblePictograms` and `getHeight`. As this is landscape-mode, they line up horizontally. Therefore we divide the width into a section for each requested visible pictogram. Then a pictogramScaling-method is called, refer listing 4.3, which makes the pictograms as large as possible, within the section they are allowed fill. The method takes the original pictogram, the limiting factor and a filter. We do however not use filters. If the pictogram is larger than 100 in width or height, it is scaled down to around 100x100. Thereafter, it scales the pictogram up or down to match the assigned section.

```
1
2  protected Bitmap pictogramScaling(Bitmap realImage, float maxImageSize,
       boolean filter){
3
4      if(realImage.getWidth() > 100 || realImage.getHeight() > 100) {
5          float ratio = Math.max((float) realImage.getWidth() / 100, (float)
               realImage.getHeight() / 100);
6          realImage = Bitmap.createScaledBitmap(realImage, Math.round((float)
               realImage.getWidth() / ratio), Math.round((float) realImage.
               getHeight() / ratio), filter);
7      }
8
9      float ratio = Math.min((float) maxImageSize / realImage.getWidth(), (
           float) maxImageSize / realImage.getHeight());
10
11     int width = Math.round((float) ratio * realImage.getWidth());
```

```
12      int height = Math.round((float) ratio * realImage.getHeight());
13
14      Bitmap newBitmap = Bitmap.createScaledBitmap(realImage, width, height,
            filter);
15      newBitmap.setDensity(0);
16      return newBitmap;
17  }
```

Listing 4.3: Scaling method for pictograms

At this point, the pictogram is added to the `PictogramView`. `PictogramView` is a class taken from the Sekvens, which creates rounded edges on the pictogram. Then, the `PictogramView` is added to the before mentioned `LinearLayout` called `mainLayout`, which contains all the pictograms to display.

The last thing we do is to scale the layout-parameters of the `ScrollView` and `mainLayout`. We define the width as: `pictogramToDisplay.getWidth()* numberOfVisibiblePictograms`, and define the height as: `RelativeLayout.LayoutParams.MATCH_PARENT`. The `pictogramToDisplay` is a regular pictogram that is scaled the same way as those in a sequence, which we use to get a pictogram width. We define these layout-parameters for the `ScrollView` and the `mainLayout`, and then center everything in the `mainLayout` by calling `mainLayout.setGravity(Gravity.CENTER)`, refer listing 4.4.

```
1 limiter = Math.min(getWidth() / numberOfVisibiblePictograms, getHeight());
2 Bitmap pictogramToDisplay = pictogramScaling(pictogram, limiter, false);
3 pictoView.setImageFromBitmap(pictogramToDisplay);
4   mainLayout.addView(pictoView);
```

Listing 4.4: Adding the scaled pictogram to the PictogramView

The `VerticallyScroll` is made the same way, but here it is the height which is divided into section. In figure 4.6 and 4.7, portrait-mode is displayed with two different settings. It scales the pictograms accordingly, and limits the `ScrollView` when the pictograms are not filling the entire height.

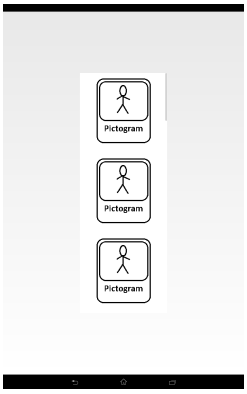


Figure 4.6: A sequence displayed with 3 visible pictograms in portrait-mode

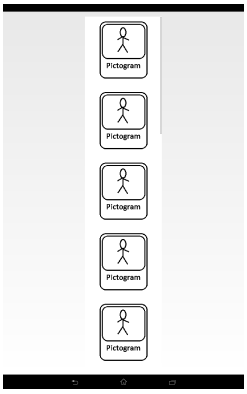


Figure 4.7: A sequence displayed with 5 visible pictograms in portrait-mode

ID	Issues	Es. hours
48	Sequenceviewer: possible settings for up to 7 pictograms	16 hours

Table 4.11: Issue ID 48

This issue involved the previous issue with dynamic show seen in table 4.10. Because the dynamic show had the desired variability, there had to be a setting to tell Sequenceviewer how many pictograms to display. This issue can, as well as issue 47, be found in appendix D. The issue is solved by receiving an intent from the application calling Sequenceviewer, with the desired number of visible pictograms. This can be seen among other intents, in listing 4.5.

```
1 //intent fra caller-app (Parrot/Zebra/Tortoise)
2 sequenceId = extras.getInt("sequenceId");
3 landscapeMode = extras.getBoolean("landscapeMode");
4 numberOfVisibiblePictograms = extras.getInt("visiblePictogramCount");
5 type = extras.getString("callerType");
```

Listing 4.5: Intent from the calling application

ID	Issues	Es. hours
50	Sequenceviewer: an arrow highlighting method	16 hours

Table 4.12: Issue ID 50

As stated in appendix D, it was a requirement that the users can mark how far they are in a sequence. This has been implemented by setting up a picture of an arrow, which are placed above the first pictogram in a sequence.

In landscape_mode.xml, an `ImageView` is created with the picture of an arrow as background. It is located in a `LinearLayout` which is placed in the hierarchy at same level as the `ScrollView`. The `ImageView` has its visibility set to *invisible*. This can be seen in listing 4.6.

```
1 <LinearLayout
2     android:id="@+id/land_arrow_linlay"
3     android:orientation="horizontal"
4     android:layout_width="wrap_content"
5     android:layout_height="wrap_content">
6
7     <ImageView
8         android:id="@+id/arrow"
9         android:layout_width="match_parent"
10        android:layout_height="match_parent"
11        android:background="@drawable/arrowtemp"
12        android:visibility="invisible"/>
13
14 </LinearLayout>
```

Listing 4.6: A section of the xml-code in landscape_mode

We then created a method in the `HorizontallyScroll` class called `setupMarker`. This method aligns the marker above the first pictogram in the sequence, and sets its visibility to *visible*. The alignment is done by first targeting the `LinearLayout` which contains the `ImageView` with the arrow. We then assign as much space as a pictogram for width and `wrap_content` for height, and define these as the layout parameters for this `LinearLayout`.

We then add alignment rules for the `LinearLayout`. This is done because it is placed inside a `RelativeLayout`. We use `ALIGN_LEFT, mainLayout.getChildAt(0).getId()` and `ABOVE, mainLayout.getChildAt(0).getId()` to make the arrow stay above the first pictogram in the sequence. Last we center everything inside the `LinearLayout`.

The arrow is set to *visible*, and the `ScrollView` is told to stay below the arrow with `BELOW, arrow_linlay.getId()`. The code can be seen in 4.7.

```
1  private void setupMarker(){
2
3      arrow_linlay = (LinearLayout) landscape.findViewById(R.id.
           land_arrow_linlay);
4      RelativeLayout.LayoutParams myParams = new RelativeLayout.LayoutParams(
           getWidth() / numberOfVisibiblePictograms, LinearLayout.LayoutParams.
           WRAP_CONTENT);
5      myParams.addRule(RelativeLayout.ALIGN_LEFT, mainLayout.getChildAt(0).
           getId());
6      myParams.addRule(RelativeLayout.ABOVE, mainLayout.getChildAt(0).getId())
           ;
7      arrow_linlay.setGravity(Gravity.CENTER);
8      arrow_linlay.setLayoutParams(myParams);
9
10
11     ImageView arrow = (ImageView) landscape.findViewById(R.id.arrow);
12     arrow.setVisibility(VISIBLE);
13     arrow.setTag("This_is_an_arrow_marker");
14
15     RelativeLayout.LayoutParams myMain = (RelativeLayout.LayoutParams)this.
           getLayoutParams();
16     myMain.addRule(RelativeLayout.BELOW, arrow_linlay.getId());
17     this.setLayoutParams(myMain);
18 }
```

Listing 4.7: Method for setting up the marker

The marker can be seen attached to a five pictogram long sequence shown in figure 4.8, which is displayed in landscape-mode.

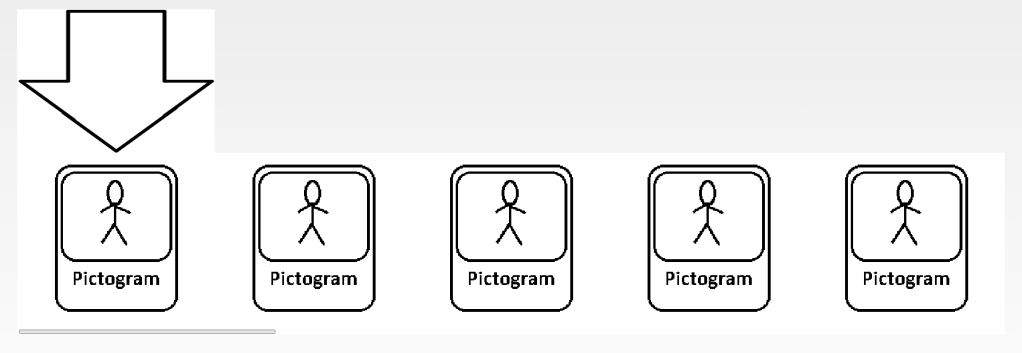


Figure 4.8: This is the arrow-marker displayed above the first pictogram in a sequence displayed in landscape-mode

ID	Issues	Es. hours
52	Sequenceviewer: show only whole pictograms	8 hours

Table 4.13: Issue ID 52

This issue comes from feedback given by the requirements group in appendix C. The feedback was an answer to questions stated in the requirements report seen in appendix B. It may be difficult for some children to see half pictograms, because they are unable to handle changes or something they do not understand.

We decided to solve this issue by using a snap-effect. This snap-effect was to be made as an extension to the already existing Android `ScrollView` and `HorizontalScrollView`. We collaborated with the GComponents group ()see section 6.4), because this functionality could be useful for other applications in the GIRAF multi-project. As can be seen in 6.4, we made a formal request for the functionality.

The solution to this issue looks as follows for HorizontalScrollView:

```
1 import dk.aau.cs.giraf.gui.GHorizontalScrollViewSnapper;
2 public class HorizontallyScroll extends GHorizontalScrollViewSnapper
```

Listing 4.8: Snap-effect for HorizontalScrollView

And the solution looks as follows for ScrollView, which is the equivalent for vertical scrolling:

```
1 import dk.aau.cs.giraf.gui.GVerticalScrollViewSnapper;
2 public class VerticallyScroll extends GVerticalScrollViewSnapper
```

Listing 4.9: Snap-effect for ScrollView

4.1.3 Remaining Issues

ID	Issues	Es. hours
36	Create snap sequenceActivity	4 hours
37	Create snap sequenceGrid	4 hours

Table 4.14: Issue ID 36 and 37

These issues springs of a requirement from the customers that children should never be allowed to watch half pictograms. With this coming to our attention, we had to change it everywhere we show pictograms. We evaluated this to take a low amount of time, but was proven to be a harder task than expected. The issue has been moved to future work (see section 7.2 in the paragraph "Snapping Scrollviews.

ID	Issues	Es. hours
51	Sequenceviewer: in settings dialog - choose what type of highlighting there should be	16 hours

Table 4.15: Issue ID 51

When issue 49 in appendix A has been resolved, and issue 50 in table 4.12 has been polished and implemented, there has be to be made a setting for preferred type of marker. This should be set up in the already existing SettingsActivity which Sekvens and Livshistorier implements. Additionally, the code needs handle this, both in Sequenceviewer and the applications calling it should use it as intent.

4.2 Sprint Evaluation

During the third sprint, a group member was forced to leave the group due to circumstances not within our control. This meant that from that point forward we had 1/4 hours less to work with. As there at this point still were no database functionality to handle sequences, there was little more to do on Sekvens. For this reason a few old, low-priority issues was re-visited and successfully closed. This included the issue of killing the application if minimized, which also took longer than

expected.

For Sequenceviewer, there was however plenty to work on. As a result, it made it to a state where it was a running as a very basic prototype, focus being foremost that it was functional. Near the end of the sprint it was announced that the database was ready to handle sequences, but at this point it was too late to change anything related to Sekvens for sprint 3. We would also have liked to do usability tests in sprint 3, but according to the Requirements group, this was not possible. This was unfortunate, because delaying it to sprint 4 meant that there would be less time to correct any potential issues found under the usability test.

Chapter 5

Sprint 4

For the last sprint, the focus of Sekvens was to get everything working with the now fully functional database from OasisLib. Furthermore we needed to implement a good way to copy and delete sequences. Being the last sprint, the main focus was however to ensure that it would work with the database.

For Sequenceviewer, the main focus was the same. It had to be functional by the end of the sprint, at the very least to a degree where sequences could be displayed. Additional features would have to be a secondary priority, as the project had a hard deadline.

5.1 Sprint Overview

The following section is a description of work made in sprint 4.

5.1.1 Sprint Backlog

ID	Issues	Es. hours
1	Syncronization with the Database	32 hours
5	Copying Sequences	16 hours
24	Delete Sequences	2 hours
25	Fix ADD/SAVE/BACK button in SequenceActivity	4 hours
33	Nested Sequences	64 hours
41	Refactor Activities	32 hours
42	Add preview Button	1 hour
44	Clean up submodules	1 hour
45	Make sure Jenkins compiles correctly with Sekvens	2 hours
54	General bugfixes	4 hours
57	Add settings	16 hours
59	Implement delete/copy with database	1 hour
64	Sequenceviewer: Display choice	8 hours
65	Sequenceviewer: Play/pause/stop sound	4 hours
66	Sequenceviewer: DB sync	4 hours
67	Sequenceviewer: Portrait-mode	8 hours
68	Sequenceviewer: Landscape-mode	16 hours

Table 5.1: This is a list of the issues we included in our sprint backlog during sprint 4

5.1.2 Resolved Issues

ID	Issues	Es. hours
1	Syncronization with the Database	32 hours
25	Fix ADD/SAVE/BACK button in SequenceActivity	4 hours

Table 5.2: Issue ID 1 and 25

This issue was the first task to make it into the product backlog for Sekvens. It was iced in the first sprint as the database was not ready at that point. At the end of sprint 3, OasisLib finished creating the models and most of the database functionality needed by Sekvens. In the beginning of sprint 4, Sekvens was then updated to reflect this change. Some of the changes has their own specific issues.

This issue covers updating all models to match the ones created by OasisLib. Specifically, Sekvens had its own representation of the classes Child, Sequence and Pictograms which were used throughout the application. Rather than reworking these to match the database, it was decided to completely remove them and use the OasisLib versions directly. This also meant updating every action taken on instances of these classes to match the possible interactions on the OasisLib versions.

Sekvens previously had its own way of storing data with its own implementation, effectively isolating it from any common database. This code was completely removed as it was no longer needed.

Fixing the database also finished the task regarding add/save/back buttons in `sequenceActivity`.

ID	Issues	Es. hours
41	Refactor Activities	32 hours
44	Clean up submodules	1 hour
54	General bugfixes	4 hours

Table 5.3: Issue ID 41, 44 and 54

Although some refactoring was done during the first sprint, MainActivity and SequenceActivity was originally messy. Having altered the code and added new features made it even worse, since it was coded on top of the mess. Essentially the code was structured poorly; methods grew large, code was duplicated and the code was full of leftovers from both this and the previous semester. To fix this, both activities had a full makeover.

One of the worst cases was the `onCreate` methods from the two big activities. These are the entry points of the activities. Even after reworking the code as much as possible, onCreate was a few hundred lines of code in both activities. At the end of the refactoring, they were boiled down into smaller methods describing exactly what happens when the activity is launched. This made the code feel cleaner, and it became easier to work with. An example for SequenceActivity can be seen in 5.1. Before, code with no apparent correlation was ordered randomly into the `onCreate` method. In the improved version everything has been sorted and relocated into relevant methods with self-explaining names.

```
1 protected void onCreate(Bundle savedInstanceState) {
2     super.onCreate(savedInstanceState);
3     setContentView(R.layout.activity_sequence);
4
5     //Make Activity killable
6     activityToKill = this;
```

```
7
8       loadIntents();
9       loadProfiles();
10      loadSequence();
11      setColors();
12      setupFramesGrid();
13      setupButtons();
14      setupTopBar();
15  }
```

Listing 5.1: The onCreate of SequenceActivity after refactoring

Along with splitting the code into relevant methods, a few cases of gathering code also happened. For example in SequenceActivity, the creation of buttons were split; Some were created and handled directly in `onCreate` while some had a method created for them. These were gathered into one method called `setupButtons` which handled everything related to buttons. Another example was `updateSequences` and `loadSequences` which functionality overlapped to a degree where they were just put together.

The refactor also included creating proper comments for the code. At the beginning of the semester there was very few comments in the main activities. Since the code was already messy, this made it hard to figure out what was going on. In spite of the refactoring, it was decided to make sure that everything was well-explained anyway. It would have been a good idea to do this to the entire code-base, but due to a lack of time, this was not possible.

Lastly, variables and method names were renamed to make them more intuitive and reflect exactly what happened within them. This was especially relevant due to the refactoring which slightly changed the functionality of some methods.

During the project a few bugs had appeared along with new features. Since we were looking the code thoroughly through, it was logical to fix these at the same time. Examples of such fixes was a crash if no pictures was returned from PictoSearch, an adapter for choices not forgetting the previous choice displayed in it and a leaked dialog box that was not closed properly.

Related to the refactoring, submodules were cleaned up at the same time. From the beginning of the project, Sekvens had several seemingly unused projects as submodules. At this point we were fairly sure that they were in fact not used. As such, PictoSearch, Ambilwarna and CategoryLib were removed as submodules. Through testing, everything was still functional. Why they were there to begin with is unknown. Removing these submodules helped clean up the code we had to manage, as any errors present in their applications would be displayed in our code editor.

ID	Issues	Es. hours
5	Copying Sequences	16 hours
24	Delete Sequences	2 hours
59	Implement delete/copy with database	1 hour

Table 5.4: Issue ID 5, 24 and 59

From the report last semester, it was stated that future work could include a copy feature to copy sequences from one child to another. This would however not have made sense to implement before having the database ready, and was thus postponed to the last sprint. The feature was implemented as a new button in the overview. When clicking the button, a pop-up window would appear where the user can press the sequences to copy. Pressing one will then move from a grid on the left (holding all sequences) to a grid on the right part of the screen. Pressing the "Copy and send to" button below will then display a GComponent, displaying all children belonging to that guardian, enabling the user to pick N children to copy M sequences to at the same time. This is illustrated in figure 5.1 and figure 5.2.

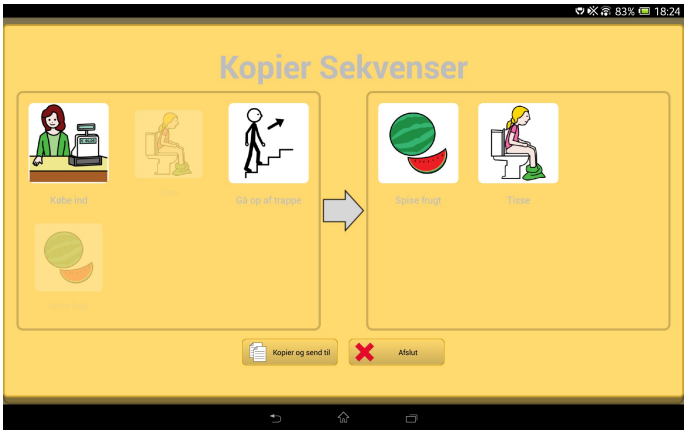


Figure 5.1: An example of a user trying to copy 2 different sequences

Figure 5.2: An example of a user trying to copy sequences to 3 different children at the same time

At the beginning of the semester it was actually possible to delete sequences, although these sequences were not part of the database. It was however very easy to accidentally delete sequences as a delete button was placed on every sequence when logged in as a guardian. There was no confirmation of deletion either. Therefore it was chosen to implement the delete functionality similar to the copy feature. A button was added which would create a pop-up window. Here, the user can press the sequences to delete, and press the delete button to perform the action. This means the user would have to mis-press two times to accidentally delete a sequence.

The reason the pop-up window was designed with the dual-grid was because a drag-and-drop functionality was intended. Unfortunately there was a major look-over in that dragging already was used for scrolling in the list of sequences, making it impossible to have both drag functionalities. Due to time constraints, the view was not altered, but was implemented to register a user press instead. If time had allowed it, a better solution could have been to have one big grid where one could add checkmarks to the sequences one wanted to delete or copy. This unexpected issue also meant that these issues went a few hours over the expected estimates.

After closing the issue it was discovered that the implementation uses an excessive amount of memory. This means that working with over 25-30 sequences at a time can crash the application with an `OutOfMemoryException`. As this task was only closed in sprint 4 however, there was not time for a redesign.

ID	Issues	Es. hours
33	Nested Sequences	64 hours

Table 5.5: Issue ID 33

This is a reopened issue from sprint 2, and is described at table 3.9. As mentioned in the referenced section database synchronization was not completely implemented, so the time spent on this assignment at this sprint is refactoring it to save it. The `sequence` model given by OasisLib had the list of frames, and these frames could have an int called `nestedSequence`. This made it possible to easily use `Frame` as the link between the sequences. This also means that it is possible to nest sequences forever, and even circle nestings. Furthermore it is possible to alter/delete sequences nested in other sequences -meaning you indirectly alter the main sequence.

ID	Issues	Es. hours
42	Add preview Button	1 hours

Table 5.6: Issue ID 42

Guardians stated that they needed a way to quickly view a sequence. This would require returning to overview, pressing the log out button, choosing a child with the sequence and then opening it. For this reason it was decided to implement a preview button in SequenceActivity. The implemented functionality prompts the guardian that the sequence will be saved and then continues to display the sequence in Sequenceviewer. The reasoning behind saving the sequence is to make it compatible with a call to Sequenceviewer. Sequenceviewer is launched with an `intent` containing the ID of the sequence to display. This is something given through the database, for which reason it must be saved. It would not be able to pass Sequenceviewer the sequence directly through an `intent`, as only simple types are supported. Thus, the easy solution was to ask the user to save the sequence first. This behavior can be seen in 5.3.

Figure 5.3: Prompting the user to save sequence before displaying it

ID	Issues	Es. hours
45	Make sure Jenkins compiles correctly with Sekvens	2 hours

Table 5.7: Issue ID 45

From previous semesters it had been decided to use Jenkins to check whether applications compiles at any given time. Except from a very minor addition to the gradle build file, the only change needed to make Sekvens build successfully was covered in 5.3 by removing submodules. One of these did not compile properly, which consequently meant that Jenkins registered the same for Sekvens, given that it was registered as a submodule.

ID	Issues	Es. hours
55	Rearrange fix	8 hours

Table 5.8: Issue ID 55

When synchronizing with the database as described in 5.2, one feature was not possible to convert successfully. Previously, rearranging was handling by using the local sequence class. The OasisLib version of the class however did not have an equivalent method. Another issue, discovered from this was that frames within sequences had no logical order - The display was simply the order in which they were found and returned by the database.

To solve both this and make rearrange work, the frames themselves were altered. Due to other groups requirements, frames had the ability to store within them a `posX` and a `posY` integer. To handle ordering of frames in a sequence, every frame would now be assigned a posX, which would then be its placement in the sequence. This also did that we could rearrange frames by changing the positions accordingly. In fact, OasisLib created a method for us to do just that.

ID	Issues	Es. hours
56	Handle no child from Launcher	8 hour

Table 5.9: Issue ID 56

At the end of sprint 3, it was decided by GIRAF to change the way logging in to the Launcher works. It would now be possible to enter the launcher without having a `childId` set, and Sekvens needs a childId to run. This means we had to check whether or not the `childId` was equals to 0, and force the user to select a child if none is chosen. When Sekvens is opened without a child, a `GProfileSelector` pops up with all the children connected to the guardian. This behavior can be seen in 5.4.

Figure 5.4: Guardian must pick a child to continue

ID	Issues	Es. hours
57	Add settings	16 hours
60	Settings Intents	2 hours

Table 5.10: Issue ID 57 and 60

To manage customer requests to display sequences differently for children, a SettingsActivity was created with two options: The orientation of the screen and how many pictograms to display at a time.

The database had no support for saving settings. This forced us to implement the settings using Android's `sharedPreferences` which is able to store preferences between user sessions [5]. This means that while the preferences are persistent on the tablet they were altered, it is not synchronized with the database at any point.

The settings are saved when the guardian closes the SettingsActivity seen in 5.4. They are uniquely saved for each child as the key used to save/fetch the settings includes the ID of the child,

which from the database should be guaranteed unique. How the preferences are saved can be seen in 5.2. When needed, they can be loaded using Android's `getSharedPreferences`. This is done when calling Sequenceviewer, and the preferences are sent through an `intent` as seen in 5.3.

Figure 5.5: The SettingsActivity - settings are dependant on the child

```
private void saveSettings() {
    SharedPreferences.Editor editor = settings.edit();
    editor.putInt("pictogramSetting", pictogramSetting);
    editor.putBoolean("landscapeSetting", landscapeSetting);
    editor.commit();
    }
```

Listing 5.2: Saving settings for a child

```
sequenceGrid.setOnItemClickListener(new AdapterView.OnItemClickListener() {
  @Override
  public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long
      arg3) {
    ((PictogramView) arg1).liftUp();
    assumeMinimize = false;

    //Load Preferences
    SharedPreferences settings = getSharedPreferences(SettingsActivity.class
        .getName() + Integer.toString(selectedChild.getId()), MODE_PRIVATE);
    int pictogramSetting = settings.getInt("pictogramSetting", 5);
    boolean landscapeSetting = settings.getBoolean("landscapeSetting", true)
        ;

    //Create Intent with relevant Extras
    Intent intent = new Intent();
```

```
14      intent.setComponent(new ComponentName("dk.aau.cs.giraf.sequenceviewer",
            "dk.aau.cs.giraf.sequenceviewer.MainActivity"));
15      intent.putExtra("sequenceId", sequenceAdapter.getItem(arg2).getId());
16      intent.putExtra("landscapeMode", landscapeSetting);
17      intent.putExtra("visiblePictogramCount", pictogramSetting);
18      intent.putExtra("callerType", "Zebra");
19      startActivityForResult(intent, 2);
20      }
21    });
22  }
```

Listing 5.3: Loading settings and putting them as intents for Sequenceviewer

ID	Issues	Es. hours
58	Add icons to all buttons	1 hour

Table 5.11: Issue ID 58

As the semester was coming to an end and we had to make the program customer ready, we had to get rid of all the placeholders and add the real icons. The assignment itself was very trivial as it was a matter of changing all of the drawables. Furthermore we wanted to add text to each button as some of the icons were a bit too bland by themselves. As we added text we noticed pictures on buttons scales from width, so if we wanted the button to be quadratic, it was impossible to add text. We decided buttons were more appealing if had quadratic buttons and no text, than rectangular buttons with text. The differences are illustrated in figure 5.6 and figure 5.7

Figure 5.6: This is the top bar before the addition of icons

Figure 5.7: This is the top bar after the addition of icons

ID	Issues	Es. hours
61	Profileselector Button	2 hour

Table 5.12: Issue ID 61

It became a requirement from the customers to be able to change children within many of the applications. We added a `GProfileSelector` to a `GButton` which made it possible to switch between the children, but not between guardians. This way it is possible easily navigate through different children without having to go all the way out to Launcher every time.

ID	Issues	Es. hours
62	Send sequence to email	4 hour

Table 5.13: Issue ID 61

This issue was a last minute decision to implement. During the fourth sprint both Livshistorier and Sekvens performed usability tests, and all of the clients praised Livshistorier ability to send sequences through e-mail. The feature was requested Sekvens as well, and a small cooperation emerged. The code existing in Livshistorier was practically copied, and slightly adjusted to match the existing code in Sekvens.

ID	Issues	Es. hours
64	Sequenceviewer: Display choice	16 hours

Table 5.14: Issue ID 64

When looking through the sequence fetched (see issue 66 in table 5.16) we attach a `onClickListener` to be able to display a choice. This is done by calling the method `attachChoiceListener` to the pictogram placeholder for the choice. The placeholder may be a pictogram from the database, or it may be a temporary pictogram. In the current version of Sequenceviewer we use the pictogram shown in figure 5.8 as choice placeholder.

Figure 5.8: The current temporary choice-placeholder pictogram. This pictogram is used, unless another is defined in the fetched sequence.

The `attachChoiceListener` method attaches a click functionality to the choice holder pictogram, as well as define what should happen `onClick`. `onClick` will call the second method used for displaying choice called `showChoiceDialog`. This method takes two parameters, the view that was clicked on and a reference to a `Map<Integer, List<Integer>>` which contains the pictograms to display in the choice. See issue 66 in table 5.16 to see how this `Map` is created and filled.

The `showChoiceDialog` method creates a new instance of the ChoiceDialog class, defined internally in the HorizontallyScroll and VerticallyScroll classes. ChoiceDialog takes 3 parameters in the constructor, the context of the `onClick` view, the `Map` with pictograms in the choice, and the `onClick` view. It then proceeds to show the choiceDialog, when it is instantiated.

The `ChoiceDialog` class defines the functionality inside the `choice_dialog`.xml-file. The xml-file is created much like landscape- and portraitmode, with a top-bar with text and a ScrollView below. We started off by inflating the choiceDialog, which displayed the choiceDialog as pop-up above the Sequenceviewer activity. We then target the LinearLayout in the `choice_dialog`.xml-file, because it is here we insert our pictograms. We also have to target the pictograms for this specific choice, as there may be multiple, therefore we use `onClick` view and use its id to point to the right position in the `Map`. Then we iterate over the pictograms in the choice and insert views in the choiceDialog. Just like in the dynamic show (see issue 47 in table 4.10) we scale the pictograms to fit the window. Lastly we attach `onClickListeners` to every pictogram in the choiceDialog, because whichever pictogram may be clicked on, has to be returned to the position of the choice placeholder in the original sequence. We use the `attachReturnPictogram` method which takes three parameters, the `onClick` view, the choice placeholder view and the choiceDialog. The code can be seen in listing 5.4, it uses same functionality as the code in listing 4.4.

```
1 public ChoiceDialog(Context context, final Map<Integer,List<Integer>>
    pictogramMap, View choiceHolder) {
2   super(context);
```

```
3   this.SetView(LayoutInflater.from(this.getContext()).inflate(R.layout.
        choice_dialog, null));
4
5   LinearLayout choiceDialogLayout = (LinearLayout) this.findViewById(R.id.
        GDialog_hsv_linlay);
6   PictogramView tempView;
7   PictogramView temp1View = (PictogramView)choiceHolder;
8   List<Integer> pictogramArray = pictogramMap.get(temp1View.getId());
9
10  for(int i = 0; i < pictogramArray.size(); i++) {
11    tempView = new PictogramView(getContext(),16f);
12    tempView.setId(i);
13    pictogram = helper.pictogramHelper.getPictogramById(pictogramArray.get(i
          )).getImage();
14    int limiter = Math.min(getWidth() / numberOfVisibiblePictograms,
          getHeight());
15    Bitmap pictogramToDisplay = pictogramScaling(pictogram, limiter, false);
16    tempView.setImageFromBitmap(pictogramToDisplay);
17
18    choiceDialogLayout.addView(tempView);
19
20    attachReturnPictogram(tempView, choiceHolder, this);
21  }
22 }
```

Listing 5.4: The constructor of ChoiceDialog sets up the pictograms and attaches onClickListeners

The ctattachReturnPictogram method targets two views, the `PictogramView` which the customer must have clicked on, and the choice placeholder view. It then removes the choice placeholder view from the sequence, because now it is no longer a choice. It then removes the `PictogramView` from the choiceDialog, and inserts it in the sequence at the position where the choice placeholder view was positioned before. Last, it dismisses the choiceDialog. The code can be seen in listing 5.5.

```
1  private void attachReturnPictogram(View v, final View choiceHolder, final
       ChoiceDialog dialog){
2    v.setOnClickListener(new OnClickListener() {
3      @Override
4      public void onClick(View v) {
5        PictogramView temp = (PictogramView) v;
6        PictogramView temp2 = (PictogramView) choiceHolder;
7        mainLayout.removeView(temp2);
8        LinearLayout mainLay = (LinearLayout) temp.getParent();
9        mainLay.removeView(temp);
10       mainLayout.addView(temp,temp2.getId());
11
12       dialog.dismiss();
13     }
14   });
15 }
```

Listing 5.5: Attaches the onClickListeners, that ends the choice by removing the choice-placeholder and inserting the pictogram that was clicked on

The functionality of a choice in Sequenceviewer can be seen in figure 5.9, 5.10, and 5.11. The child first sees the sequence, then presses the choice and chooses which activity to perform, and then perform to complete the rest of the activities in the sequence.

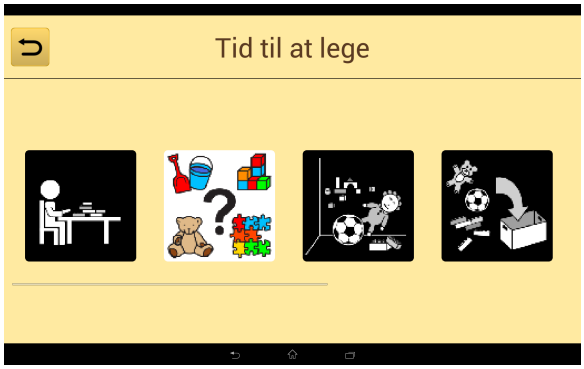


Figure 5.9: A sequence in Sequenceviewer with the 2nd pictogram containing a choice

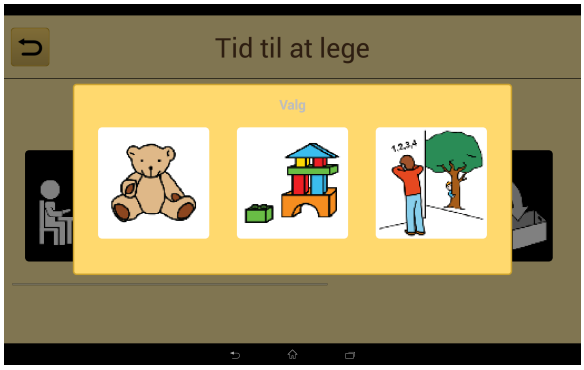


Figure 5.10: Now the choice has been clicked on, and the three pictograms to choose among are being displayed

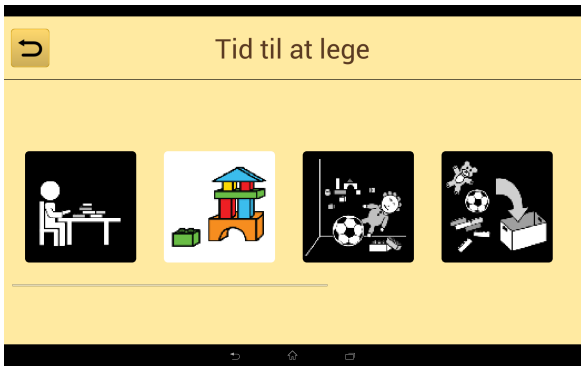


Figure 5.11: The middle pictogram in the choice has been selected, and is now being displayed where the original choice-pictogram were placed

ID	Issues	Es. hours
65	Sequenceviewer: Play/Pause/Stop sound	32 hours

Table 5.15: Issue ID 65

This issue comes from the general specification requirements, see appendix D. Additionally, it is a requirement from the PiktoOplæser group to be able to play sound, otherwise their application would not have any use of Sequenceviewer.

The functionality for playing sound has already been developed by the PiktoOplæser group together with the PictoCreator group. They created a library in the pictogram-lib project, which holds the functionality. Therefore, Sequenceviewer uses this library to play and stop sound. There

was no request from the customers for pause, neither did any of the other two groups implement the functionality.

To use the library, we implemented three methods: `onClickPlaySound()`, `playNext()`, and `soundDonePlaying()`. We use a list of byte-array to hold the sound for each pictogram in a sequence. That makes it possible to iterate over the pictograms in a sequence, and display they attached sound accordingly. This happens by instantiating a version of a PictoMediaPlayer from the pictogram-lib, as well as implementing the CompleteListener, which tells the PictoMediaPlayer when a pictogram is done playing the attached sound. These initiating parts can be seen in listing 5.6.

```
1 import dk.aau.cs.giraf.pictogram.PictoMediaPlayer;
2
3 public class HorizontallyScroll extends GHorizontalScrollViewSnapper
    implements CompleteListener{
4 ...
5 PictoMediaPlayer pictoMediaPlayer = new PictoMediaPlayer(getContext());
6 ...
7 }
```

Listing 5.6: Instantiating the PictoMediaPlayer from pictogram-lib

We start off by setting up buttons for playing and stopping sound. This is done in a method called `setupSoundButtons()`. We defined buttons for playing and stopping sound in the landscape_mode.xml-file, and we now set their visibility to *visible*. We attach an `onClickListener` to each of the buttons, and add functionality for playing sound and stopping sound. The code for the `onClickListener` can be see in listing 5.7.

```
1         playButton.setOnClickListener(new OnClickListener() {
2             @Override
3             public void onClick(View v) {
4                 try {
5                     onClickPlaySound();
6                      index = 0;
7                 }
8                 catch(Exception e) {
9
10                }
11            }
12        });
13
14        stopButton.setOnClickListener(new OnClickListener() {
15
16            @Override
17            public void onClick(View v) {
18                pictoMediaPlayer.stopSound();
19                index = 0;
20            }
21        });
```

Listing 5.7: onClickListeners for playSound() and stopSound()

Now the pictoMediaPlayer goes through the sequence and playing the sound attached to the pictogram. This happens by initiating the playSound by starting `onClickPlaySound()`, which sets the customListener to the current pictoMediaPlayer. It then calls `playNext`, which set the datasource of the pictoMediaPlayer to be the first sound element to be played in `soundToPlay`. It plays the sound and increments the index pointer. Once the sound is done playing, `soundDonePlaying`

calls `playNext()` again, and if the index within the number of `soundToPlay` it continues, until this does not hold. The code for these three methods can be seen in listing 5.8.

```
1     private void playNext(){
2         if(index < soundToPlay.size()){
3             pictoMediaPlayer.setDataSource(soundToPlay.get(index));
4             pictoMediaPlayer.playSound();
5             index++;
6         }
7     }
8
9     private void onClickPlaySound(){
10        pictoMediaPlayer.setCustomListener(this);
11        playNext();
12
13    }
14
15    @Override
16    public void soundDonePlaying() {
17        playNext();
18    }
```

Listing 5.8: The three methods associated with playing sound in Sequenceviewer

Figure 5.12 shows a sequence with sound attached to the pictograms. The sound can be played by clicking the play-button. The sound can be stopped by pressing the stop-button.

Figure 5.12: A sequence with the possibility to play and stop sound.

ID	Issues	Es. hours
66	Sequenceviewer: DB sync	32 hours

Table 5.16: Issue ID 66

Even though section 6.1 explains the collaboration about common database functionality, there were misunderstandings in how to use it. This resulted in Sekvens using the database slightly different than Livshistorier. Additionally, PiktoOplæser did not have time to implement the database at all, therefore they are not part of the database sync of Sequenceviewer. To asses this issue with different database-usage, Sequenceviewer implements two different ways of fetching a sequence with pictograms from the database.

In `MainActivity` we get the `sequenceId` as intent from the application calling Sequenceviewer. We use this id to get the sequence from the database by using the helper created by the OasisLib

group. The OasisLib group also created a library with models, which contains a sequence class. We import their two classes, create an instance of the sequence class and use the helper to get the sequence and pictograms from the database, according to the `sequenceId`.

```
import dk.aau.cs.giraf.oasis.lib.Helper;
import dk.aau.cs.giraf.oasis.lib.models.Sequence;
...
private static Sequence sequence = new Sequence();
...
sequence = helper.sequenceController.getSequenceAndFrames(sequenceId);
...
```

Listing 5.9: Importing functionality from database-group and fetching a sequence from the database

In the constructor of HorizontallyScroll and VerticallyScroll, we take as argument the type of the application calling Sequenceviewer. We assign them a number as in the following order:

1. Sekvens
2. PiktoOplaeser
3. Livshistorier

The result of the collaboration in making the database-schema can be seen in appendix E.

Sekvens and Livshistorier share some similarities when using the database. They both store a Sequence as the schema defines it, both use Frames to store every pictogram/choice/nested sequences, in the database. But the difference arises in how to use the Pictogram_collection.

Sekvens uses the database by only storing pictograms in the Pictogram_collection, whenever they want to store a choice, e.g. multiple pictograms for only one section in a sequence. They store a single pictogram in the Frame by setting the pictogram_id in the Frame. Sekvens stores a choice by setting the pictogram_id in the Frame, to be the first pictogram in Pictogram_collection, and storing all the choice pictograms in Pictogram_collection.

Livshistorier uses the database by always storing pictograms in Pictogram_collection, a single pictogram or multiple. They allow the pictogram_id in the Frame to be a placeholder for the pictogram they want to use when displaying a choice. This means that this pictogram_id can become null.

For Sekvens, we start off by looking into the Pictogram_collection, if this collection contains no elements we know that it is a single pictogram, and we get that from the Frame. If the Pictogram_collection contains more than 0 elements, we know that it is a choice, we therefore look in the Frame for the pictogram_id, e.g. the placeholder for the choice. Last, we fetch all the desired pictograms for the choice from the database and store them for later display. The code can be seen in listing 5.10. The choiceFlag is used to attach an `onClickListener`, refer issue 64 in table 5.14.

```
if(callerApp == 1) {
  if(sequence.getFramesList().get(i).getPictogramList().size() == 0) {
    pictogramId = sequence.getFramesList().get(i).getPictogramId();
  }
  else if(sequence.getFramesList().get(i).getPictogramList().size()>0){
    choiceFlag = true;
    if(sequence.getFramesList().get(i).getPictogramId() != 0) {
      pictogramId = sequence.getFramesList().get(i).getPictogramId();
    } else {
      tempChoicePic = true;
    }
    for(int x = 0; x < sequence.getFramesList().get(i).getPictogramList().
        size(); x++) {
      tempArray.add(sequence.getFramesList().get(i).getPictogramList().get(x
          ).getId());
```

```
14      }
15      choiceMapping.put(i,tempArray);
16    }
17
18  }
```

Listing 5.10: Fetching a sequence from Sekvens using the database

With Livshistorier, we look at the size of the Pictogram_collection. If the size is 1, we know they want to display a single pictogram. If the size is larger than 1, we than they want to display a choice, and we set up the same functionality as with Sekvens. We then simply take out all the pictograms and store them for later display. The code can be seen in listing 5.11. Notice we only look at the pictogram_id in the Frame, for the choice-placeholder pictogram.

```
1  else if(callerApp == 3) {
2    if(sequence.getFramesList().get(i).getPictogramList().size() == 1 ) {
3      pictogramId = sequence.getFramesList().get(i).getPictogramList().get(0).
           getId();
4    }
5    else if(sequence.getFramesList().get(i).getPictogramList().size() > 1) {
6      choiceFlag = true;
7      if(sequence.getFramesList().get(i).getPictogramId() != 0) {
8        pictogramId = sequence.getFramesList().get(i).getPictogramId();
9      }
10     else {
11       tempChoicePic = true;
12     }
13     for(int x = 0; x < sequence.getFramesList().get(i).getPictogramList().
           size(); x++) {
14       tempArray.add(sequence.getFramesList().get(i).getPictogramList().get(x
             ).getId());
15     }
16     choiceMapping.put(i,tempArray);
17   }
18 }
```

Listing 5.11: Fetching a sequence from Livshistorier using the database

The functionality for fetching and displaying nested sequences has not been resolved. The database-schema contains a nested_sequence_id in the Frame-table, it just has to be fetched when it is possible to display it.

ID	Issues	Es. hours
68	Sequenceviewer: Landscape-mode	2 hours

Table 5.17: Issue ID 68

Sequenceviewer got working functionality to display sequences in landscape-mode. It has implemented the features developed as other issues and works without failures. Therefore this issue has been resolved.

5.1.3 Remaining Issues

ID	Issues	Es. hours
67	Sequenceviewer: Portrait-mode	2 hours

Table 5.18: Issue ID 67

Portrait-mode is set up in the VerticallyScroll class. The code is very similar to the one in HorizontallyScroll, but there are minor issues remaining before the features developed in other issues are implemented correctly. Portrait-mode has a problem with scaling the pictograms up, making them too large for the dynamic show to work properly. Before Portrait-mode can be considered resolved, this problem has to fixed.

The problem with scaling can be seen in figure 5.13. The settings are set to display five pictograms in a sequence, but the scaling is clearly not working as intended.

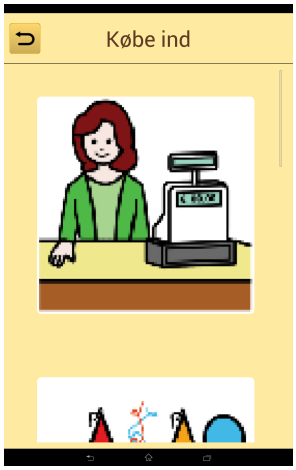


Figure 5.13: A sequence displayed in portrait-mode, with the setting set to 5 visible pictograms. The scaling is clearly not working

5.2 Usability Test

In sprint 4 it was possible to arrange a usability test for Sekvens. This was arranged with help from the Requirements group. The usability test created the possibility to identify if the development had changed Sekvens positively, and locate usability problems or missing features.

5.2.1 IDA - Instant Data Analysis

For the usability test it was decided to roughly use the model of Instant Data Analysis (IDA). IDA has the benefit of being very time-efficient while still highlighting many issues [6].

The idea with IDA is to have a test subject perform given tasks on the software, and log everything at the same time. An alternative approach would be to record everything, transcribe it and analyze it later on. This is however very time expensive and exactly what is avoided here. Instead, an analysis session is scheduled immediately after the test to create a list of discovered issues.

IDA normally requires four people with each their unique role:

Test subject A person who attempts to perform tasks on the software to test. The subject is asked to think aloud to clarify choices made when using the software, providing the feedback

needed.

Test monitor Interacts with the test subject in terms of introduction and informing of the tasks the test subject should perform. Sits next to the test subject, but does not help with the tasks. Can ask questions for clarification if the test subject is not clear on the basis of an action.

Data logger Logs any noteworthy behavior from another room while the test subject performs the assigned tasks to the software. The test subject is usually monitored through a camera.

Facilitator Manages the analysis session and helps identify and categorize issues.

5.2.2 Test Setup

For this usability test, it was decided not to have a facilitator. We found that the tasks of the facilitator could be distributed between the test monitor and data logger. Furthermore, the suggested setup for the data logger was not possible. In lack of proper monitoring options, the data logger sat in the same room as the test subject and test monitor.

After the test, we did an analysis session to create the final list of issues that was found during the test. This list can be found in 5.2.3.

The list of tasks to be performed by the test subjects was defined beforehand in a way that makes the test subject use every aspect of Sekvens. Test subjects were informed that we would like them to think aloud during the test and that the test monitor would not help them perform the tasks. Furthermore, since the use of Sekvens included various fictive users, test subjects were informed about the name of their profile and the list of children they had access to administer.

The list of assignments can be found in Appendix F . The test were performed on two test subjects. It should be noted that due to an unforeseen database change, the test unfortunately differed between the two tests. This means that the fictive names was different and that available pictograms had changed as well. The tests were the same, but program data was different.

5.2.3 Test Results

Both test subjects had never used the application before, although knowing about the project in general. This means we got an insight as to how intuitive and user-friendly the program is for new users.

Beside the test we actually got a request when finishing up. One test subject would like to be able to send sequences to email. This originated in the Livshistorier application being able to do this. This is seperated from the test because it is not an issue with the program but a request for new functionality.

Under the test, several issues were highlighted. These has been categorized within the following categories:

- C: Cosmetic/minor issue such as a wrong icon or a bug with little interference.
- S: Severe difficulty completing the assigned task. Takes 30 seconds or more to find a functionality.
- F: Critical error. User is not able to complete the task and is unable to proceed without assistance.

These are the issues found in Sekvens:

C: Misleading icon A visual issue immediately recognized by the second test subject was that the icon for the exit button was a red X. We were informed that this usually means something is canceled, and suggested to find a different icon. As this is an icon from GIRAF_Components that

several groups use, this would be preferred changed from there. An alternative icon could be a man going out through a door.

C: Choosing multiple pictograms at the same time One of the test subjects attempted to insert a nested sequence instead of choosing multiple pictograms. This could possibly be solved easily by renaming "pictogram" to "pictogrammer" (pictograms) in the add dialog. This would highlight that it is possible to insert more than one pictogram.

C: Identifying if settings/copy is saved One test subject was unable to see if the sequence she had copied and the settings she had altered were actually saved. This could easily be solved by displaying a dialog box informing of the result. The test subject however correctly assumed changes were indeed saved.

S: Locating Nested/Choice feature Both test subjects had issues locating these features and generally took a long time finding them. Both test subjects however stated that after using it once, it was quite intuitive. An alternative could be to discard the dialog and instead add a seperate button for each feature. This would however also create some cluttering as there are already 6 buttons in the activity.

One test subject had a hard time locating the choice button because she mistook it for a settings or special menu. A possible fix could be to find a better naming for a choice in a sequence.

It was suggested by one test subject that the placeholder for a choice should be a question mark similar to Livshistorier.

S: Identifying a choice or nested sequence Both test subjects were unable to directly identify which pictogram in a sequence was a choice or a nested sequence. One of them were able to find it by trial-and-error while the other simply reached the conclusion that it could not be seen. This was however expected as identifying placeholders had not been implemented at this point.

S: Locating the copy/delete button One test subject had difficulties locating the copy and delete buttons. Once finding them however, she had no idea why she did not see them in the first place, but emphasized that it was very intuitive. From observation, she was not in doubt she had found the right buttons once seeing them. Rather than attempting to fix this issue, we attribute it to be an unfortunate oversight by the test subject. Having more test subjects could help identify if it really is an issue.

5.2.4 Test Analysis

The usability test turned out to display some interesting results. Some issues were expected as the implementation was not entirely done. This was clearly displayed when none of the test subjects could identify the type of pictograms from each other (Nested sequence, pictograms and choices).

On the less obvious side some of the behavior was very unexpected. A few examples of this is was a test subject attempting to drag pictograms down to make them into a choice, as well as the other test subject expecting that picking multiple pictograms would automatically make it a choice.

It was at that point we realized that the test subjects had a bias. The overall setup arranged by the requirements group involved one customer testing all applications on the same day. Test subjects expected all applications to work in similar ways whenever they seemed similar. While explaining some of the unexpected behavior, it also highlighted another issue: Consistency across applications. While the applications share components such as GComponents and OasisLib, this does not mean that things are done the same way. Unifying similar procedures across applications could help improve the overall feel of the project and make it much more intuitive.

Some of the issues were definitely local to Sekvens, such as the issue to identify whether or not a change has been saved, and the issue of identifying the type of a pictogram. But for a solution to inconsistencies across applications, cooperation with the responsible groups would be needed.

5.3 Sprint Evaluation

This sprint, Sekvens had some major refactoring done, which greatly helped readability. This was found to be needed, given the amount of code added to the application. After the refactor, the application was also easier to work with, speeding up development.

At the point of the deadline, Sekvens was fully functional, in that all features was working with the database. It was possible to copy, delete, create and edit sequences from within Sekvens. It was however also a turbulent finish to the sprint as some dependencies pushed breaking changes on the last day. This ultimately meant that a few workarounds had to be implemented in Sekvens to meet the deadline with a fully functional version. This is however not something that will affect end users, and is only noticeable within the code. Sequenceviewer was especially under pressure, and as expected, some features could not be implemented. It was however functional and usable for all groups that had requested to use it.

Chapter 6

Collaboration

With GIRAF being a multi-project, it often made sense to work across groups. While all applications had their own purpose, many were connected in some way through the same dependencies or similar functionality. This provided options for collaborations where all participants could profit from it, and improve their own project for less effort. This chapter describes the collaborations Sekvens has been a part of.

6.1 Database Planning

When we were assigned this project, the database system did not fully support saving sequences of pictograms. The database was built to save only pictograms to profile, and you had to create code to interpret which pictograms were part of what sequences. This was not ideal and required a lot of work to use. Additionally, it required users to search through all pictograms every time you had to open a sequence. We gathered with all of the groups who had interest in saving sequences in the database. The groups were the following Livshistorier, PiktoOplæser and the iOS group. When collaborating to find a joint database system for groups with different needs, we needed a set of requirements from each groups. We arranged a meeting and discussed the different requirements and created design conventions for each user story, thereafter deciding what the minimal common we could reach was.

Each group were to present an ER-model or idea for a database that entailed all of the features that were needed for the minimal database design. The draft we came up with are portrayed in 6.1:

As profiles and pictograms already exists in the database, these tables are not further described. We add a new sequence table that contains all of the information needed to describe a sequence from all of the groups, there is nothing really questionable here. The second table, Frames, is where the most of the magic happen. The way the pictograms/nested sequences/choices are saved in the database are that they are saved to each frame. A frame can store one or more pictograms, or a nested sequenceId. This way when you are fetching a sequence from the database, you search the list of frames for frames matching the sequenceId and setting them up according to posX.

The draft we submitted was quickly decided as a good baseline, but Livshistorier refined it by removing the choice table, as this was possible to create through the pictogram table. We agreed that Livshistorier would collaborate with OasisLib on converting the draft to an official ER-diagram.

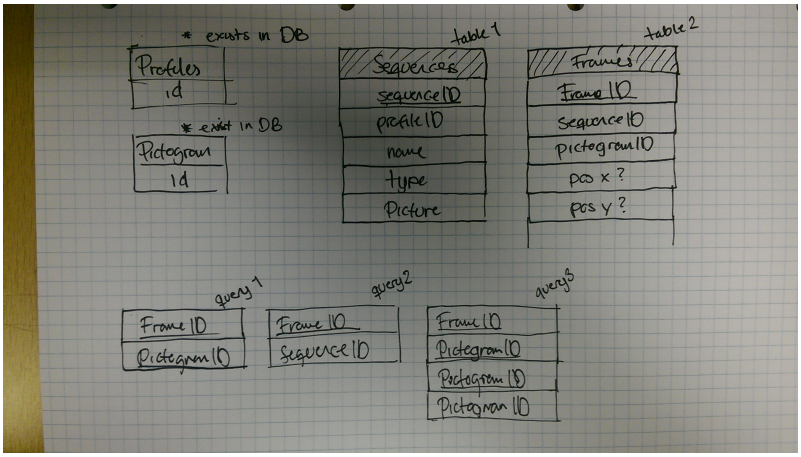


Figure 6.1: Our idea of a potential database for sequences

6.2 SequenceViewer

The GIRAF multi-project contains three applications which share a common feature, which is sequences of pictograms. The three applications are Sekvens, Livshistorier, and PiktoOplæser. Sequenceviewer was created during a meeting dedicated to database planning, but there was a common need for displaying pictograms, therefore Sequenceviewer was created. The Seqeuenceviewer was decided to be a front-end application, which the three other applications should call whenever they want to display a sequence to a child. This new application should ensure a common interface whenever the children interact with the system. The sequences of pictograms will be displayed in the same way, whichever application the guardians previously used to make a sequence of pictograms. The purpose of Sequenceviewer is to act as a service to the three other applications. It is not suppose to be apparent to the guardians or the children, that they actually enter Sequenceviewer. The position of Sequenceviewer in the GIRAF multiproject is shown in figure 6.2.

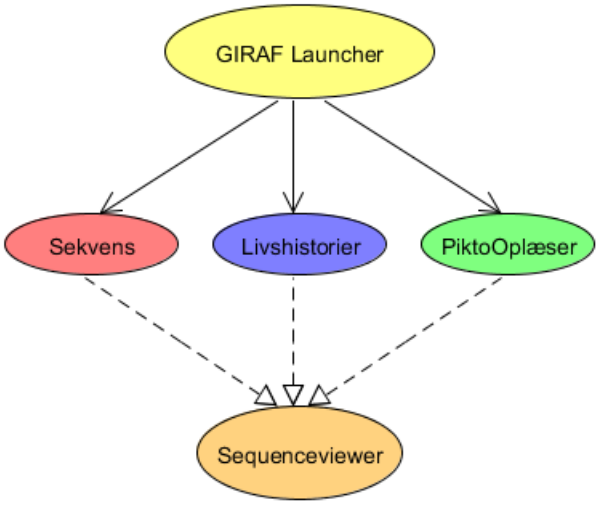


Figure 6.2: The position of Sequenceviewer in the GIRAF multiproject.

The stippled lines between the three applications and Sequenceviewer, represent the fact that it should not be apparent to the users that they enter Sequenceviewer.

Here are the requirements to Sequenceviewer, these became clear during the meeting:

- It should be able to show a choice between 2 or more pictograms. Choosing a pictogram

should now take the place in the sequence, which before held the choice-pictogram.
- It should be able to show nested sequences, meaning that a pictogram in a sequence can also be a whole sequence. When a child presses the sequence it should open the sequence related to it.
- It should be able to mark a pictogram as "done" by highlighting or moving an arrow-marker.
- It should show a sequence of pictograms up to 7 at a time. This should be adjustable in a settings menu for each of the applications before Sequenceviewer is opened.
- It should be able to be display a sequence in landscape- or portrait-mode. This should also be a choice in a settings activity.
- It should be able to exit Sequenceviewer and return to the previous activity.
- It should not show pictograms in the sequence with cut edges or even half pictograms.

6.3 Code Review

This section covers a two-way code review in collaboration with the CAT group. Both groups did a review of the Main Activities, the entry point for the applications.

For a collaborative part, it was chosen to perform a code review between sw609f14 (Cat) and sw607f14 ('Sekvens'). According to Wikipedia [7], code reviews can be done in several different ways, e.g. pair programming or informal walkthroughs. We adopted recommendations 2, 3, 4, and 7 from Cohen's practices [8]. We also applied the way "Foredecker" meet up again after reading through the code and discuss it [9]. Our primary focus was to keep it not too long, as to be able to maintain focus, as Cohen suggests in point 3 in his article.

Our goal with the code review was to get other people to look our code through, and find obvious errors the groups had stared themselves blinds on, as well as the general quality and readability of the source code, and if the structure of the source code - methods, classes, names - was meaningful.

Group sw607f14's finding in Cat's code are listed in section 6.3.2, and group sw609f14's findings in 'Sekvens' are listed in section 6.3.3. How the feedback was used, is described in section 6.3.5 for 'Sekvens', and section 6.3.4 for Cat.

6.3.1 Common Traits for Sekvens and Kategoriværktøjet

It was found that the code in general had good readability. Both method and variable names made sense in the context. The code was generally split up well into proper methods. The code was well commented, explaining components that were not self-explanatory.

6.3.2 Sekvens' corrections to Kategoriværktøjet

Duplicated code Throughout the code certain variables were cleared several times, creating some redundant code. It was found that a better alternative could be to create a method to clear these variables rather than doing it manually every time. The same applied to updating some of the graphical components. These updates could be done in a single method rather than several times throughout the code.

Lazy class A minor issue was found that Cat overrode Android's `onPause` method. The only action taken within this method was however to call Android's `super.onPause`, meaning that the overridden `onPause` did nothing that would not have been done automatically in the Android lifecycle.

Contrived complexity / Complex conditionals In a few cases, unnecessary checks were performed. One case was an if-sentence that checked if a variable was empty. If it was indeed empty, it would re-create that variable as empty. Another case was excessive use of if-sentences

where the checks could either be combined or alternatively be set up using a switch. While the code is generally easily readable, this could give some of the code even better readability.

Another case of contrived complexity was assigning all properties of an element to another, rather than copying the entire element. A case was also found where an instantiated variable was instantiated again for no apparent reason.

Other A few examples were found where the code was perfectly functional, but technically wrong, not used as intended or unnecessary. An example of wrongly used code was the use of controllers. Rather than using the helper class from Oasis, these helpers were circumvented by instantiating new controllers. Another minor detail was to cancel a window rather than dismissing it. While not changing any functionality, it would have been technically correct to use Android's dismiss method.

Adapters were used a lot in the Activity and were fully functional. In the code however, it turned out that every time the data for an adapter was changed, a new adapter was created on that data. The correct way would have been to use Android's `notifyDataSetChanged` method to update the existing adapter.

Lastly a few potential bugs were found. Some images were not centered as intended and it was possible to avoid selecting a child by using the Back button, which would create further issues in the application.

6.3.3 Kategoriværktøjets corrections to Sekvens

Well structured and good use of methods Generally the code made and edited by group sw607f14 were well structured and simple to read and understand. It had a good and appropriate use of methods, which made it easier to get an overview of the class. Furthermore these methods were of appropriate size.

Grouping of variables Variables and their declaration could have been grouped better, such that related variables were placed together. This way it would have been easier to find certain variables, when trying to find a variable.

Nested classes with some inappropriate intimacy The class `MainActivity` had two nested classes, which depended on variables from `MainActivity`. These could be moved into separate files and made independent from the `MainActivity` to avoid inappropriate intimacy among the classes.

Some excessive use of literals instead of variables When accessing other applications through Zebra, hardcoded literals where used to send data to the starting application. These literals could be made into variables, such that multiple equivalent literals could be avoided.

6.3.4 Changes made to Kategoriværktøjet

Several changes were made to the class `MainActivity` as a result of the feedback from sw607f14.

Adapters In several places we created and set new adapters to update the GUI. This was changed to use the `notifyDataSetChanged` method instead. In some places however, this did not work, and we were thus forced to set the adapters again.

Controllers Instead of instantiating our own instances of the necessary controllers, we switched to use the `Helper` class from Oasis. This made some lines a bit longer, but was otherwise worth it. In the process, we also removed all usage of CategoryLib in `MainActivity`.

Dialogs Instead of calling `cancel` on dialogs when they were closed, we now call `dismiss`.

`onPause` method As the `onPause` method was unneeded, it was removed. It had been used before when Cat was using XML files to store its data, but only the code handling this had been removed.

Contrived complexity and complex conditionals As was noted, we had some cases of complex conditionals. The `OnActivityResult` method was the worst in this. Before refactoring the method, it consisted of several if-statements (some of which made a "backwards" check, e.g. `if (isIcon != true) { ... }`). After refactoring, the method contained a switch statement, with four cases, which is much easier to read.

Rejected suggestions The potential bug where it was possible to avoid selecting a child by using the Back button, was a general issue involving the Giraf-Component `GProfileSelector`, and can best be fixed by the group working on GUI. Therefore it was unfortunately not handled this year, but should be looked at next year.

Other suggestions for improving the code was rejected simply due to a lack of time.

6.3.5 Changes made to Sekvens

As a result of the CAT groups constructive feedback, the MainActivity of Sekvens were changed.

Moving code to XML Some code was highlighted which could be set in XML rather than be done programmatically during runtime, and we did exactly that, as it were preferred setting elements in XML whenever possible.

Try-Catch It was noticed by CAT that several `try-catch` were present without any real functionality; Errors would be caught but not dealt with. Most of the `try-catch` present in the code was not needed at that point, and were simply removed as it was not possible for the exceptions to occur.

Multiple classes within same file As we had two other classes in the same file as the MainActivity class, it was suggested to either separate them from the file or move them to the bottom of the file to separate them from methods used in MainActivity. Since the classes were minor and uniquely used in MainActivity, it was chosen to move the classes to the bottom of the file as suggested.

String literals Rather than using constants when sending or receiving intents, MainActivity had string literals, which CAT noted as a bad idea. This change was not completed in time, but is definitely a good idea.

Rejected suggestions Feedback on code style was also given. This included using `foreach` loops rather than using counters with `for` loops and sorting variables by functionality rather than type. It was also suggested to refactor a case where a method was nested within another. This was however also rejected, as we found that for the particular case, increased readability was better than moving the inner method.

6.4 Other Ad-hoc Activities

Apart from the the major collaborations, Sekvens was in a unique position where we had a lot of dependencies. As a result, our work often overlapped with the work of other groups. Below is described ad-hoc activities that took place on a day-to-day basis.

Livshistorier In GIRAF Livshistorier and Sekvens are very similar. The only difference between the two is that Livshistorier is meant to be used as a collaboration between the child and the guardian. The applications were so similar that we decided to hold meetings to talk about shared features. The two major topics of conversation were the database for sequences (see section 6.1) and the joint viewing tool (see section 6.2). Aside from these some minor points were also agreed upon, these were as following:

- We should share our codes because most of sequences functionality can be used in Livshistorier.
- The settings should be saved for each child.

The biggest cases of shared code happened when Livshistorier needed a proper way to create sequences. This resulted in borrowed code from Sekvens such as the `SequenceViewGroup`. This method is responsible for everything related to displaying a sequence being edited along with listeners to replace, delete and move pictograms.

Using the same code across applications makes it more user friendly. The user will only have to learn one way of editing sequences/lifestories rather than remembering two distinct ways. Near the end of sprint 4 the costumers really enjoyed a feature in Livshistorier that enabled the customer to send sequences as emails in order for them to print the sequence. This was ported to Sekvens as per request from the customer.

GComponents Since it was decided to share common GUI components, most graphical components in Sekvens had to be changed. Using the components from the GComponents group was made very intuitive, and we were mostly able to use them without problem. Sometimes however, we could not resolve either how to use a component properly, or how to give a component a certain functionality. This resulted in several cases where we lent one of their team members to provide assistance, or alternatively cases where new functionality had to be added to the GComponents. Furthermore, since this group did not use its own components, other than for testing, a few errors was caught in the Sekvens application, and corrected by the GComponents group.

During sprint 3, two formal request were made to the GComponents group. Sequenceviewer needed a snapping-effect for a ScrollView and a HorizontalScrollview. We talked to a member of the GComponents group about what we needed, and since nothing like it existed, two issues were added to their redmine-page. (See table 6.4.

[REQUEST] From 607 (Zebra)	[REQ] GHorizontalScrollViewSnapper: Create component
[REQUEST] From 607 (Zebra)	[REQ] GVerticalScrollViewSnapper: Create Component

OasisLib Ad-hoc activities were also done with the OasisLib, assisting in both using their classes properly and providing new functionality on request. Given that Sekvens uses OasisLib for everything database and model related to sequences, these run-ins occured often. As with the GComponents, our position meant that several bugs were discovered and later corrected by OasisLib.

GIRAF Sekvens was initially designed as a stand-alone application. In spite of the existing Launcher project, Sekvens was not set up to use it. The idea was that Sekvens should be opened

from the Launcher, sending two ID's that Sekvens would then depend on (For the Guardian/Child launching the application). This was set up during the second sprint, but also caused some issues. Just like Sekvens, Launcher was dependent on other projects that would fail from time to time, consequently also affecting Sekvens. Launcher itself also received a few changes that sometimes broke.

Other than dependency issues, Launcher implemented the functionality of changing settings from within the Launcher. Some minor collaboration went into making Sekvens settings work with the Launcher.

PictoSearch To get pictures for the sequences in Sekvens, we had a dependency for the PictoSearch application. It was our experience that the responsible group were not present very often, neither for status meetings or during the usual work hours. Unfortunately this caused some problems in that the application at all times were several revisions behind on their database (OasisLib) dependencies. Other applications like Sekvens and Launcher however updated to the newest versions whenever available. Effectively, this meant that in the best case it was not possible to get pictures through PictoSearch, as the program did not match the desired version of OasisLib. In many revisions it would just crash.

Due to the lack of presence from the responsible group, it was not possible to collaborate properly, and a working version was never created from their side.

To get around this issue, methods were implemented to create hardcoded sequences with hardcoded pictograms within Sekvens. Towards the end of the project other groups put together a working version of PictoSearch which was used instead. This version was functional with newest database and could thus be used.

Requirements In the first two sprints, the overall project had a group focusing mainly on requirements. They were to have a full requirement analysis ready at the end of sprint 1, but the report was sub-par. Sekvens is very dependent on requirements from the clients, as it is a front-end application. It was decided to create a thorough report for the requirement group, in order to get as specific requirements as possible. The report can be seen in appendix D. The ideas in the report was clarified with full hand-drawn illustrations of Sekvens, to ensure that there would be no loss of communication between us, the requirements group and the clients. It focuses mainly on interfaces of requirements they had already requested, such as illustrating how the user knows how far he is in a sequence. Examples of this were using an arrow or highlighting of a pictogram. We provided several different paper prototypes, for the various scenarios, and all the pictures can be seen in appendix B, and answers in appendix C.

Chapter 7

Evaluation

7.1 Conclusion

At the beginning of the semester, some design patterns was decided upon (See section 2.3). These patterns were followed throughout the project with one notable exception. MainActivity is no longer singleton, in that opening `nestedMode` is a second instance. When deciding on design patterns it was not considered that having a second instance for a different purpose could be relevant. The intent with the singleton pattern was rather that several instances of the application should not exist at the same time. Given that we can re-use MainActivity in `nestedMode` gave grounds for a justification of breaking the design pattern, especially given that it does not duplicate a running activity.

The goal of this semester was to finish the existing application rather than develop new. As described through the report we have continued the development on Sekvens, pushing for a state where it would be ready for a potential release of GIRAF. As it has been described from sprint 3 and forward, Sekvens was split up into a management application and the hidden Sequenceviewer.

The latter is in a state where it is functional, but missing some polishing. It also lacks the feature of somehow pointing to the current pictogram in a sequence using a marker. The request for this feature is known from the work of the requirements group. During the usability tests we however got surprisingly little feedback regarding Sequenceviewer. In fact there was only a side-question of whether it was possible to see the sequence vertically rather than horizontally, which was later seen in settings. The issue of missing indicators for choices was also ignored during the tests, another feature we have through requirements.

In other words, judging on the usability test alone, Sequenceviewer could be done within limited time. Whether the two usability tests gives a clear image of whether it is perfect is questionable. Looking aside from the missing indicators for choices and the pointer to pictograms, we conclude that more testing is required to give a good evaluation on its state. The indication is however that while it is not finished, it is headed in the right direction.

Sekvens had much groundwork done when we started on the project. This allowed us to develop requested features and polish it. At the usability tests, two suggestions were made; that we use a question mark to indicate choices and the ability to send a sequence by email. The latter was implemented with help from the Livshistorier group as described in 6.4. Indicators were however not done in time. From the usability tests we gathered that while some features were problematic to find, it was possible, even for a beginner to use the application. This is very positive, and indicates that the design is intuitive. It was not ideal to have the usability tests in the last sprint as this meant very little room to develop any requests or fix any issues. The fact that only one request could not be developed in time could be an indicator that Sekvens is in a good state. This

does however not mean that it is ready for release. We find that the missing indicators of choices and nested sequences is something that should not be left out. Furthermore, there are several minor issues that could be improved upon, as described in 7.2. If GIRAF would be developed on for another semester, we would recommend that Sekvens to be worked on for an other 1-2 months.

Lastly, it should be emphasized that two usability tests can not give a clear answer as to the state of the projects. Further usability testing would be required for a full analysis of the projects. What can be concluded is that functionality- and feature-wise, Sekvens is close to a release version.

7.2 Future Work

In the following section we will discuss the future work of Sekvens. Aside from the big functional improvements listed below, there are a lot of optimizing and improvements to the general flow of the project. The general flow could be improved through more usability testing and applying whatever improvements they suggest, as well as fixes to missing dialog boxes, misinterpreted buttons or unclear instructions.

Nested Sequences as mentioned at the section nested sequences (see 5.5), there is a couple of different issues with the implementation chosen. As mentioned the biggest problem seem to be that it is possible to alter/delete sequences as the main sequence simply references the nested sequence. This is a problem, because deleting the nested sequence will first make the application crash. Later, when a new sequence is made with an ID equal to what the deleted nested sequence used to have, the main sequence will link to a wrong sequence. There is various different ways to resolve this issue, and one of them could be creating a new 'hidden' sequence which is only linked to through the main sequence. The downside of this solution is the amount of memory used to store 'hidden' sequences which in themselves offer nothing to the application.

The other problem with nested sequences is the possibility to create circular nesting, meaning sequence #1 can be nested into sequence #2 while sequence #2 is nested into sequence #1. This is not really a problem while running the app, seeing it does not crash the guardian version of the application. The problem occurs when the child is supposed to watch the sequence. Keep in mind that Sequenceviewer does not fully support nested sequences yet, but the way we intended Sequenceviewer to open nested sequences was to open a new instance of Sequenceviewer once the nested sequence is reached. This way it would be possible to support forever nested sequences, not limiting the application. If Sequenceviewer was implemented like this, circular nesting would cause the application to keep calling itself every time the nested sequence was reached thus never ending. Somehow the application must prevent circular nesting, and the easiest way would probably be to when `setupNestedMode` is called through `sequenceActivity`, and a sequence is chosen you could save the `sequenceId` in a temporary list of `Integers`. Then the next time `setupNestedMode` is called, you could exclude all of the sequenceIds already in the `List`.

Copy and delete dialog boxes The functionality is already completely there, it is possible to both copy and delete sequences. The problem is the `GGridView` provided by GComponent along with the `PictogramView` created in Sekvens uses a lot of memory. This arises when you mark a lot of sequences, the application takes way to much memory and often run out of memory resulting in crashes. This is quite a big problem as the application crashes, which the customer emphasized should bea voided. There is not really an easy solution to this, the solutions are either optimize `GGridView` and `PictogramView` or alternatively figure out a less memory dependent method to copying or deleting sequences.

Snapping Scrollviews This problem is a bit more coding dependent. It was a requirement listed by the costumers, see appendix D, that the children were never exposed to half pictograms

as they might be confusing. This means every place that uses `PictogramViews`, needs to implement a snapping grid ensuring that only full pictograms are shown. This is the case in the overviews `GGridView` and the `SequenceViewGroup` in `sequenceActivity`. In `sequenceActivity` it might be pretty arbitrary to do it as it always shows 4 `PictogramViews` horizontally in 1 row and scale them according to the width of the screen. It might be harder in the `MainActivity` as it scales according to width but stacks rows as sequences are added.

Implement Sequenceviewer with PiktoOplæser During this semester, PiktoOplæser did not manage to implement the database functionality finished during sprint 4. Therefore, Sequenceviewer did not have the possibility to implement their way of using the database, and setting up functionality for their application. Next semester, cooperation should finish this absence of functionality. From the Sequenceviewer perspective, code should be written for their database-usage, and provide the PiktoOplaeser-group with information and help with setting up the intents and call to Sequenceviewer.

Miscellaneous The following is a list of minor changes which does not take long to fix, but was big enough to receive a notable mention.

- `isInEditMode` is an ancient relic of the past, a variable that used to be from times when `sequenceActivity` was used to both display and edit sequences. This boolean is completely unnecessary now, and could be removed from the entire project.
- When entering Sekvens without a `childId` being sent from Launcher, the `GProfileSelector` pops up and you need to pick a child. It is right now possible to press the hardware back button. The problem is that the Profile Selector is a view and not an activity, so you cant simply override `OnBackPressed`.
- When copying sequences there is no confirmation that the sequences have been copied. This could be achieved through a dialog box or simply closing the entire window completely.
- When altering settings, there is no save button. It simply saves every time back is pressed, giving no confirmation the settings are saved. This is a simple problem which could be solved through a save button or a confirmation dialog box when pressing back.
- The exit button needs to be invisible in `NestedMode`.
- Indicators for choices and nested sequences, when editing and showing a sequence. This could be achieved through a mark on the `PictogramView`.
- Right now the `GGridView` in `MainActivity` always shows 5 `PictogramViews`. This should be dynamic such that the amount of sequences shown are determined by the width of the screen.

7.2.1 Rejected tasks in Product Backlog

Over time, the Product Backlog was cluttered due to tasks that had for various reasons been invalidated. This could happen due to new issues, new knowledge about an issue or change of focus. Therefore the backlog was looked through, and the following issues has been removed from it.

ID	Issues	Es. hours
7	Marked pictogram not highlighted	8 hours
8	Child mode needs to be more intuitive	32 hours
9	Sequence does not look clickable	16 hours
22	Portrait mode	32 hours

Table 7.1: This is a list rejected after reevaluation

7.2.2 Remaining Product Backlog

Even after sorting through the backlog at the end of the last sprint, some issues still remained. This was both tasks that have never been picked up and tasks not completed in sprints. The following is the final product backlog. This can be used as reference for anyone wanting to continue development on Sekvens.

ID	Issues	Es. hours
2	Updating a picture in sequence ->; no update in overview	8 hours
13	Resizing delete buttons on pictograms	4 hours
16	It is possible to make identical sequences for the same child	8 hours
23	SequenceGrid needs to be dynamic	8 hours
36	Create snap in SequenceActivity	4 hours
37	Create snap in SequenceGrid	4 hours
46	Improve user handling	4 hours
49	Sequenceviewer: Highlighting with a outline	16 hours
51	Sequenceviewer: In settings dialog - choose what type of highlighting there should be	2 hours
53	Sequenceviewer: Blur pictures out of focus	8 hours
63	Sequenceviewer: Display nested sequences	16 hours
69	Sequenceviewer: Marker drag over to next pictogram	16 hours
70	Sequenceviewer: Marker stays at pictogram when scrolling the sequence	8 hours
71	Sequenceviewer: Marker snaps to pictogram	16 hours
72	Sequenceviewer: 2 rows/columns. 1 with a marker, 1 with a sequence	16 hours
73	Sequenceviewer: Display 2 rows/columns	16 hours
74	Sequenceviewer: While playing sound, follow the pictogram which has sound being played	32 hours

Table 7.2: This is a list remaining for Sekvens and Sequenceviewer

7.3 Reflections

The customers, possible end users and collaboration across groups made this semester very interesting. It was very different from anything we had tried before, but the feeling is that it overall went well. Furthermore, this was the first time we had a large codebase at the beginning of a semester. In previous semesters, any code had been created from scratch. It was interesting to try gaining an overview of other students code, learning exactly what and how code was done.

Since this is the first time we have collaborated across groups, there was certain unexpected issues with management. At the beginning of sprint 1, we had little idea how many dependencies there were for Sekvens, even if it was not connected properly to all of them. Dependencies, and the work associated with managing them took much effort. This sometimes allowed for less time to be spent on Sekvens itself. Under the circumstances, it went quite well. But if we were to do it again, it would have been ideal to create a debugging mode from the start to include some hardcoded data (persons and sequences). This could have saved some time, when stuck because of a failing dependency. Given the lack of experience with multi-projects, we were however unable to predict it.

Sharing one tablet across two groups also had unforeseen problems. During the problems it would vary how far ahead we were. This could in the worst case mean that the two applications working in the group room had dependencies to different versions of the database. It was sometimes possible to access two tablets, but the best solution had been to have at least one per group. This

was also due to slow emulators which could take as long as 5 minutes to start up.

The overall experience with this semester has been good. As a group we have become better at both programming and management across groups.

Chapter 8

Bibliography

[1] "Agilewrap." http://www.agilewrap.com/index.html.

[2] "Design patterns." http://sourcemaking.com/design_patterns.

[3] "Giraf: Zebra," student report, Aalborg University, 05 2013.

[4] "Android activity." http://developer.android.com/reference/android/app/Activity.html.

[5] "Sharedpreferences." http://developer.android.com/reference/android/content/SharedPreferences.html.

[6] "Instant data analysis: Conducting usability evaluations in a day." http://delivery.acm.org/10.1145/1030000/1028050/p233-kjeldskov.pdf.

[7] "Code review." http://en.wikipedia.org/wiki/Code_review.

[8] J. Cohen, "11 proven practices for more effective, efficient peer code review." http://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review.

[9] "How do you perform code reviews?." http://stackoverflow.com/questions/310813/how-do-you-perform-code-reviews.

Part I

Appendix

Appendix A

Product Backlogs

The following shows the product backlog as it evolved during the project.

A.1 Sprint 1 - Product Backlog

ID	Iced	Issues	Description	Es. hours
1	x	Syncronization with the Database	Work together with the Database group to correctly link Sequence with the database - making extracting and saving images easier.	32 hours
2	x	Updating a picture in sequence ->; no update in overview	When opdating a picture in sequence, the picture is not shown in the overview.	8 hours
3		Cancel and replace action function	Being able to mark a pictogram cancelled. Futhermore being able to replace pictograms in a sequence.	12 hours
4		Destroy Sekvens if minimized	When pressing the home button on the tablet, it should destroy the application, and not pause it. For example it would be confusing, to open from paused state, instead of opening into the overview.	8 hours.
5		Copying Sequences	Being able to copy sequences to other children incase of similar activities.	16 hours
6		Changing application name	Change the name from Zebra to something else.	1 hour
7		Marked pictogram not highlighted	When the pictogram is clicked it does not highlight.	8 hours

8		Child mode needs to be more intuitive	Child mode relies on scrolling right now - we need to make it more intutive e.g. marked activities/pictograms as done. Overall the flow of child mode needs to be reworked.	32 hours
9		Sequence does not look clickable	It is hard to see that you can click a sequence and choose to "open" it.	16 hours
10		Changing a sequence name is not intuitive	When naming a sequence it is hard to see that you can click on the "space" to name it.	8 hours
11		Analyzing TODOs in code	The TODO's are located in: Child (class), MainActivity and SequenceViewGroup - Note there can be several TODO's per file.	16 hours
12		Make add pictogram button placement intuitive	The "+" sign is wrongly placed.	16 hours
13		Resizing delete buttons on pictograms	The size of the deletebutton have to scale proportionally with the width of the PictogramView.	4 hours
14	x	Import temporary pictures into Sequence	Right now, no pictures are visible in the program right now.	4 hours
15		Original drag location flickers when rearranging	When dragging the location dragged from flickers when the animation is done. Comment from last semester: Seems to be depend on hardware. The old galaxy tab does not have this problem. Also seems to be worsened by rounded image corners.	32 hours
16		It is possible to make identical sequences for the same child	Should be fixed, maybe as a check vs name and order of pictograms.	8 hours
17	x	Solve TODO in SequenceViewGroup - Can child be bigger than parent?	It is currently unknown whether a child can be bigger than the parent space allowed. Blocked because no images are available yet.	2 hours
18		Child selected upon opening the program is not highlighted	Child selected upon opening the program is not highlighted - starts at the first child of the list but it is not highlighted.	2 hours
19	x	Zebra is installed with two app-icons	When installing the zebra-application, two icons are created. This is a launcher dependency issue.	16 hours

20		When a sequences is created it should have a return to overview button	When a sequences is created it should have a return to overview button, right now it's still in the sequence view.	8 hours

Table A.1: This is a list of the issues we had in our product backlog going into sprint 1

A.2 Sprint 2 - Product Backlog

ID	Iced	Issues	Description	Es. hours
21		Create view-mode	Blur pictiures out of focus. Only full pictures(no half/auto adjust). Marker-choice: arrow or highlight, locked or dynamic position of marker. Possible settings for up to 7 pictograms.	128 hours
22	x	Portrait mode	The sequences should be viewable in portrait mode.	32 hours
23		SequenceGrid needs to be dynamic	Right now, SequenceGrid only allows 4 pictograms in the Grid.	8 hours
24		Delete sequences	Delete sequences requires database syncronization, with the LocalDB.	2 hours
25	x	Fix ADD/SAVE/BACK button in SequenceActivity	Currently blocked because it is not possible to save sequences (Missing database functionality).	4 hours
26		Change all buttons	The new GUI buttons are available in the wiki for GUI.	8 hours
27		Change GUI of Viewgroups	The new GUI Viewgroups are available in the wiki for GUI.	16 hours
28		Remove the Childlist	Update and adapt to context, because child is sent from Launcher.	32 hours
29		Update to Oasis	Update to the new database.	4 hours
30		Refactor overview	Create BaseOverview. Create GuardianOverview.	32 hours
31		Rework sequenceTitle box to look editable	If we can make it look clickable we can remove the button.	4 hours
32		Display some temporary sequences in overview	Since there is no database for sequences yet, we need to be able to display some temporary ones for development purposes	4 hours
33		Nested Sequences	There is a need for nested sequences, similar to daily scheduel (self-explanatory)	64 hours
34		Implement choices in sequences	Implement sequences so that they can contain the option to make choices.	16 hours

35		After adding pictograms to a sequence it should not go to the "sequenceActivity", it should go to "MainActivity"	After making a sequence, the application should load the overview, instead of loading the sequence in SequenceActivity.	8 hours

Table A.2: This is a list of the issues we had in our product backlog going into sprint 2

A.3 Sprint 3 - Product Backlogs

ID	Iced	Issues	Description	Es. hours
36		Create snap in SequenceActivity	When you scroll, it should snap back to only showing full pictograms only	4 hours
37		Create snap in SequenceGrid	When you scroll, it should snap back to only showing full pictograms only.	4 hours
38		Create nested sequence placeholder	Create a pretty placeholder (and add number of pictograms) Linearlayout: imageview + textview	2 hours
39		Create choice placeholder	Add a simple choice placeholder (write how many pictograms are in the choice) Linearlayout: imageview + textview	2 hours
40		Create exit button for Sekvens	Currently not possible to exit application without using "back" button	1 hour
41		Refactor Activities	Refactoring the entire MainActivity and SequenceActivity	32 hours
42		Add preview Button	Add a button calling sequenceviewer with the sequence we're editing	1 hour
43		Remove sequence name button	As the name suggest, remove the button as it clutters the UI	0.5 hours
44		Clean up submodules	Remove submodules we don't rely on anymore	1 hour
45		Make the project Jenkins compatible	Make sure Jenkins compiles correctly with Sekvens	2 hours
46		improve user handling	Create dialogboxes to guide the user through the application	4 hours

Table A.3: This is a list of the issues we had in our product backlog regarding sekvens going into sprint 3

47		Sequenceviewer: Dynamic show	It should be able to set a setting in Sekvens/Livshistorier/-Parrot that decides how many pictograms should be shown at a time - the functionality needs to be implemented in sequenceviewer	16 hours
48		Sequenceviewer: possible settings for up to 7 pictograms	Do not show more than 7 pictograms at once. This is a requirement from the stakeholders/requirements group	16 hours
49		Sequenceviewer: highlighting with a outline.	When the user choses a method of highlighting to be the a highlighting, it should directly show a outline for that sequence or pictogram.	16 hours
50		Sequenceviewer: an arrow highlighting method	The arrow should in a fixed position. So when the the pictogram is marked as "cancel/done", the pictograms move, but not the arrow.	16 hours
51		Sequenceviewer: in settings dialog - choose what type of highlighting there should be	Should it be an arrow or a highlighting method?	2 hours
52		Sequenceviewer: show only whole pictograms	The entire sequence should snap, so it is impossible to see half pictograms	8 hours
53		Sequenceviewer: blur pictures out of focus	When a user (child or guardian?) presses a pictogram, then blur the others out. This should be an alternative to highlighting and set in settings	8 hours

Table A.4: This is a list of the issues we had in our product backlog regarding sequenceviewer going into sprint 3

A.4 Sprint 4 - Product Backlogs

ID	Iced	Issues	Description	Es. hours
54		General bugfixes	The program is filled with temporary code fixes from when the database sync wasn't in place	4 hours
55		Rearrange fix	As the database sync has been fixed, rearranging had to entail rearranging in the database as well	8 hours

56		Handle no child from Launcher	It was decided across the entire semester, that it should be possible to log in in launcher without a child, this means the ChildID can be 0. Sekvens doesn't work when no child is selected so we have to make sure everything is handled accordingly	8 hours
57		Add settings	Saving settings for each child as they are needed in sequence-viewer	16 hours
58		Add icons to all buttons	all the old placeholders needed to be replaced with the real icons	1 hour
59		Implement delete/copy with database	Adds database functionality to fix delete and copy	1 hour
60		Settings Intents	send the settings with the call to sequenceviewer as intents	2 hours
61		Profileselector Button	adjust our relogButton to the profileselector option GUI has provided	2 hours
62		Send sequence to email	Make it possible to send a sequence to an email in order to print it out	4 hours

Table A.5: This is a list of the issues we had in our product backlog regarding sekvens going into sprint 4

ID	Iced	Issues	Description	Es. hours
63		Sequenceviewer: Display nested sequences	Should be able to display nested sequences	16 hours
64		Sequenceviewer: Display choice	Requirement from the Sequence-group and the Lifestories-group	16 hours
65		Sequenceviewer: Play/-pause/stop sound	Requirement for the Parrot-group.	32 hours
66		Sequenceviewer: DB sync	Gather all the information through sequenceID as you can not send a sequence as intent	32 hours
67		Sequenceviewer: Portrait-mode	Supporting portrait mode - reading sequences from top and down	2 hours
68		Sequenceviewer: Landscape-mode	Supporting landscape mode - reading sequences from left to right	2 hours
69		Sequenceviewer: Marker drag over to next pictogram	Being able to drag the marker to the next pictogram in a sequence	16 hours
70		Sequenceviewer: Marker stays at pictogram when scrolling the sequence	Being able to to snap the marker to a pictogram, not changing what is points to when scrolling the sequence.	8 hours

71		Sequenceviewer: Marker snaps to pictogram	Snap-effect to the marker, when you you drag marker to a new pictogram.	16 hours
72		Sequenceviewer: 2 rows/-columns. 1 with a marker, 1 with a sequence	Requirement from Lifestories-group.	16 hours
73		Sequenceviewer: display 2 rows/columns	Requirement from Lifestories-group. They wanted two sequences to be displayed, one on top of the other.	16 hours
74		Sequenceviewer: While playing sound, follow the pictogram which has sound being played	When playing sound, a marker should highlight the pictogram with sound being played at that moment, following along the sequence when playing it all.	32 hours

Table A.6: This is a list of the issues all related to sequenceviewer

Appendix B

Requirements Report

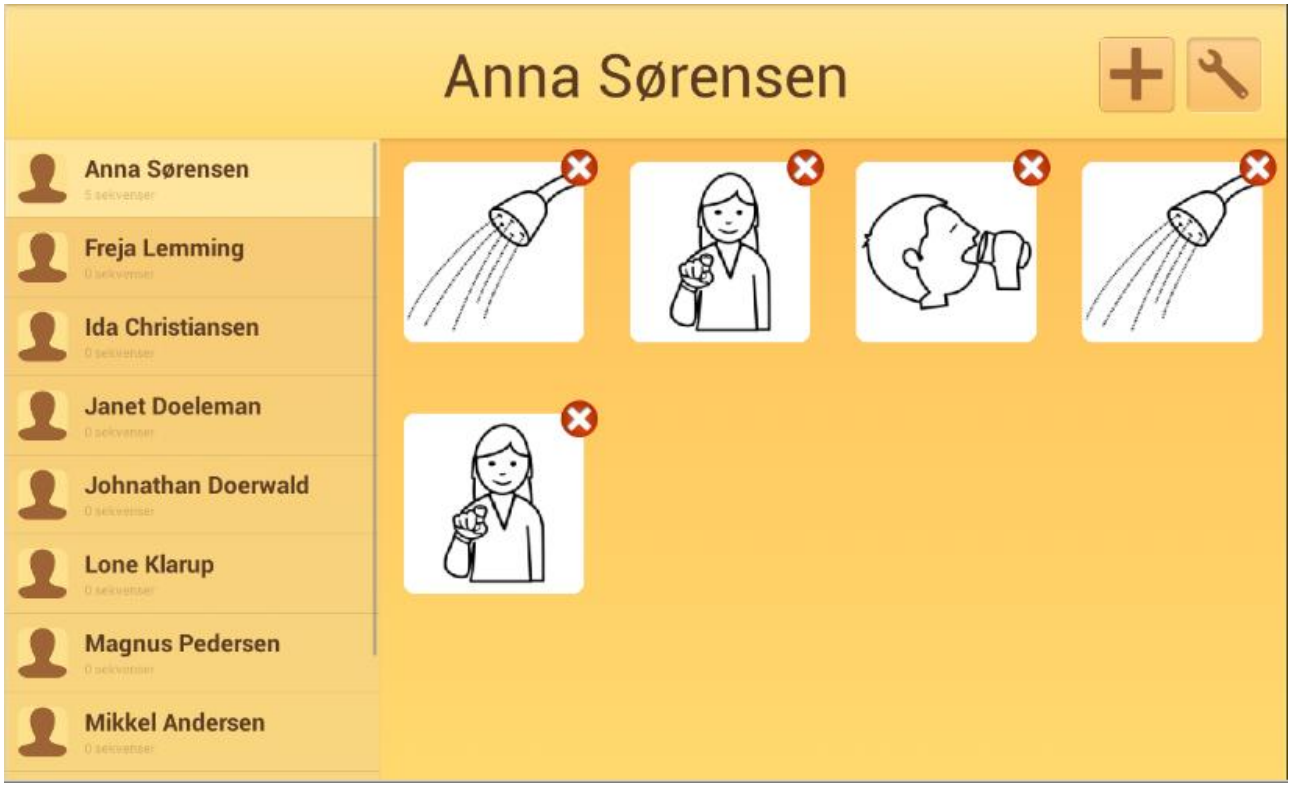
The following is the report created to help ask relevant questions for the customers. It consists of an introduction explaining exactly what the application is capable of, followed by questions and examples that would help lead to specific requirements. These especially cover how sequences should be displayed for the user when played and explores the interest for having some preferences options for the application.

Opfølgende spørgsmål til kravspecifikationsgrupperne og stakeholders. Vi er gruppen der arbejder på Zebra som nu er kaldt Sequence. Vi laver sekvenser af pictogrammer.

Vi vil gerne følge op de krav vi har fået. Der er en række krav som vi gerne vil stille opfølgende spørgsmål til, for at afgøre hvilken løsning vi skal implementere i vores application. Denne tekst er skrevet som udgangspunk til kravspecifikationsgruppen, men bør bruges i dialog med stakeholders for at præcisere hvad vi ønsker svar på. Der er lavet paper-prototypes af de løsningsforslag vi har lavet angående de opfølgende spørgsmål.

Præsentation af Sequence-applikationen, den som før hed Zebra

Vi giver først lige en kort præsentation af den application vi arbejder på, så i har et udgangspunkt til de opfølgende spørgsmål.



BILLEDE 1 - Sådan ser vores application ud når man åbner Sequences. Dette er et overview-vindue. Her er barnet Anna Sørensen valgt ud af de 8 synlige profiler. De 5 sekvenser der er bundet til Anna Sørensens profil er vist til højre. Skruenøgle-ikonet tillader at toggle "guardian-mode", hvor man kan ændre/slette sekvenser. Plus-ikonet åbner vinduet hvor man lave en ny sekvens. X-ikonet viser i dette billede, at man er i "guardian-mode", altså i edit-mode, og man kan derfor slette de sekvenser der tilhører et barn.

BILLEDE 2 - Sådan ser vores applikation ud når man har trykket på plus-ikonet i overview-vinduet. Dette er et tilføj-sekvens-vindue. Her har man den sekvens man er i gang med at tilføje midt på skærmen. Plus-ikonet med stiblet linje åbner pictoSearch applicationen hvor man kan vælge de pictogrammer man gerne vil tilføje til sekvensen. De bliver tilføjet i den valgte rækkefølge. X-ikonerne vil i dette vindue slette det valgte pictogram fra sekvensen. Flueben-knap-ikonet gemmer den sekvens man har lavet. X-knap-ikonet starter en pop-op box, der spørger brugeren om de gerne vil: ”Gem”, ”Gem Ikke” eller ”Annuller”. Man kan navngive sin sekvens med at trykke på navnet, i dette tilfælde et navnet ”Unavngivet sekvens” Unavngivet sekvens. Man kan også ændre på det pictogram man får vist for hele sekvensen i overview, dette gør man ved at trykke på det lille pictogram øverst til venstre, i dette tilfælde pictogrammet for bade.

BILLEDE 3 - Dette vindue er en sekvens som vises uden at være i ”guardian-mode” altså uden edit-mode. Dette vil være udseendet når barnet får en sekvens som barnet skal følge. Man kan ikke ændre navn, pictogrammer eller lign. Man kan godt markere et pictogram, og barnet kan også trykke på X-ikonet for at afslutte sekvensen. Dette symboliserer at barnet er færdig med sekvensen, og er i nogle tilfælde ikke noget barnet får brug for. Når man trykker på X-ikonet vil man blive spurgt om man ønsker at afslutte sekvensen, man kan svare ”Afslut” eller ”Annuller” for at blive.

Opfølgende spørgsmål om krav til Sequence-applikationen

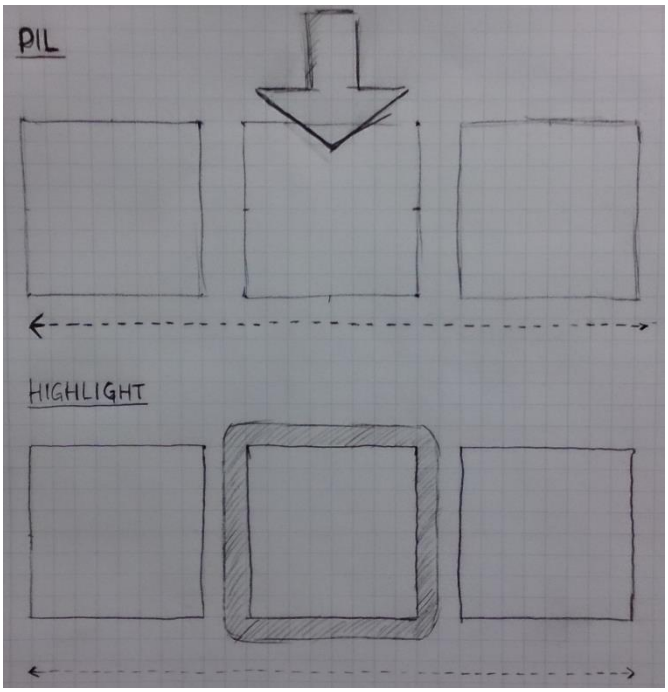
Vi har nogle opfølgende spørgsmål til valg af funktionalitet og udseende i vores applikation som vi gerne vil have stakeholders til at tage stilling til. Vi tager udgangspunkt i de udarbejdede krav udleveret af kravspecifikationsgruppen:

Krav 1-2: *”Autistic profiles must be able to mark how far in a sequence they are.” and “The solution must indicate which step in the sequence have been marked.”*

***Valg 1** – Ønskes der en pil som markør eller en omkransende markør?*

På nuværende tidspunkt er der en mulighed for at highlighte et pictogram. Vi vil gerne have stakeholder til at tage stilling til 2 nye forslag om en markør der viser hvor langt et barn er i en sekvens:

1. Skal markøren være en pil?
2. Skal markøren være et highlight(omkransende kasse)?
3. (spørg evt. stakeholder om de ønsker en helt 3. mulighed).



BILLEDE 4 – Dette billede viser 2 forskellige muligheder for at markere hvor langt man er i en sekvens.

Valg 2 *– En fastlåst markør eller en markør der kan flyttes?*

Som på Billede 4, her kunne det være muligt både at flytte på sekvensen hvor markøren var fastsat. Eller man kunne flytte sekvensen og markøren blev stående på det valgte pictogram, og hvis man så ønskede at flytte markøren skulle man sætte fingeren på markøren og flytte den individuelt.

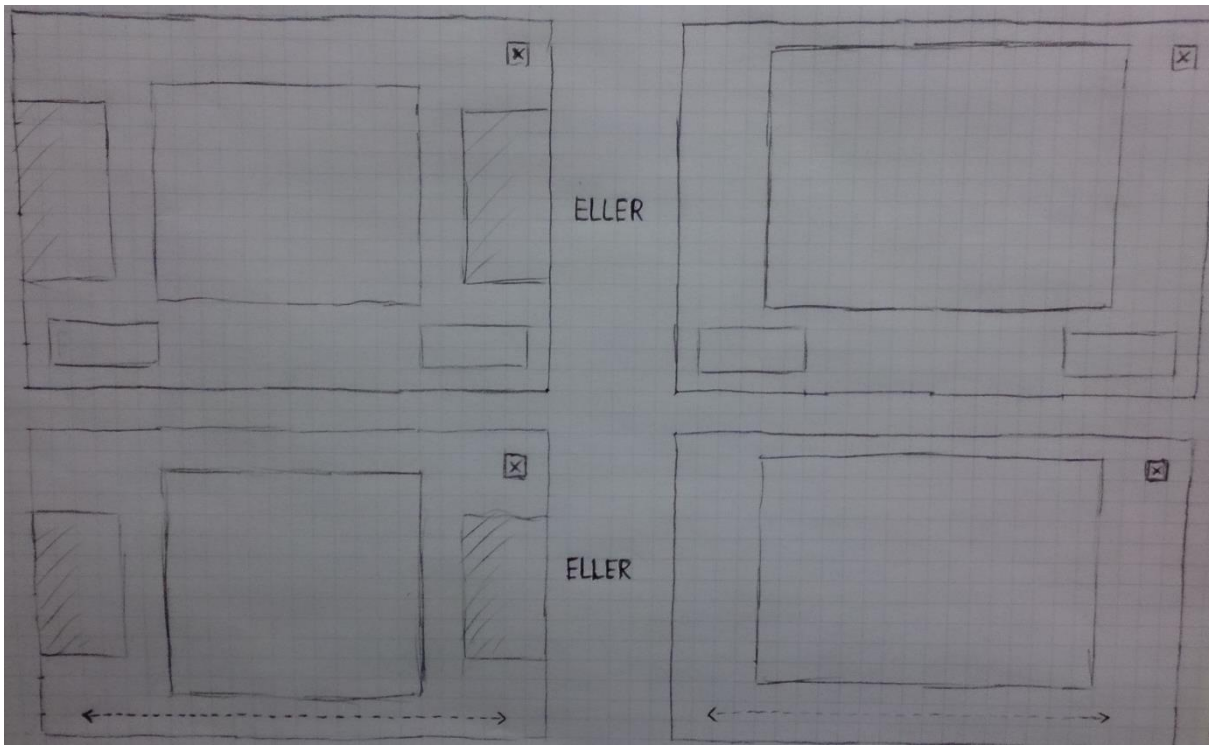
1. Skal markøren være fastsat i vinduet, så du flytter hele sekvensen?
2. Skal markøren kunne flyttes individuelt, så man både kan flytte markøren og sekvensen uafhængigt af hinanden.
3. (spørg evt. stakeholder om de ønsker en helt 3. mulighed).

Krav 3: *"There must be different ways to view the sequences, such as show three pictures at once or just one picture".*

Valg 3 *– Single-view, multi-view eller en kombination af en slags?*

Pt. er der i Sequence kun mulighed for multiview med 4 pictogrammer. Stakeholder har allerede udtrykt ønske om kun at vise et enkelt pictogram af gangen. Der er givet nogle forslag til udseende på Billede 5.

1. Skal vinduet være single-view, så man kun kan se et enkelt pictogram af gangen?
2. Skal vinduet være multi-view, så man kan se flere pictogrammer af gangen?
3. Skal der være mulighed for at vælge dette som en indstilling for hver enkelt barn?



BILLEDE 5 – Dette billede viser 2 forslag til udseende af vinduet med sekvenser i Sequence. Øverst er et multi-view og et single-view hvor man har knapper i bunden til at skifte til forrige eller næste pictogram i sekvensen. Nederst er også vist et multi-view og et single-view, men her skal man scrolle med fingeren for at skifte til næste pictogram.

***Valg 4** – Hvor mange pictogrammer i et multi-view?*

På BILLEDE 1-3 er der vist multi-view med 4 pictogrammer i hvert vindue. På BILLEDE 4, er der vist et vindue med 3 pictogrammer. På Billede 5, er der vist et vindue med 3 billeder hvor det forrige og det næste pictogram er gjort mindre.

1. Ønskes der 3 synlige pictogrammer i et multi-view vindue?
2. Ønskes der 4 synlige pictogrammer i et multi-view vindue?
3. Ønskes der X antal synlige pictogrammer i et multi-view vindue?
4. Ønskes der en grafisk effekt på de billeder der ikke er det valgte pictogram? (Som BILLEDE 5, hvor de er gjort mindre og halverede.)

***Valg 5** – Skal man scrolle til næste pictogram, trykke på en knap, eller måske begge muligheder?*

På nuværende tidspunkt kan man kun scrolle imellem pictogrammer. På BILEDE 5, kan man se der er knapper på de 2 øverste billeder, og der er en scroll-indikator på de 2 nederste billeder.

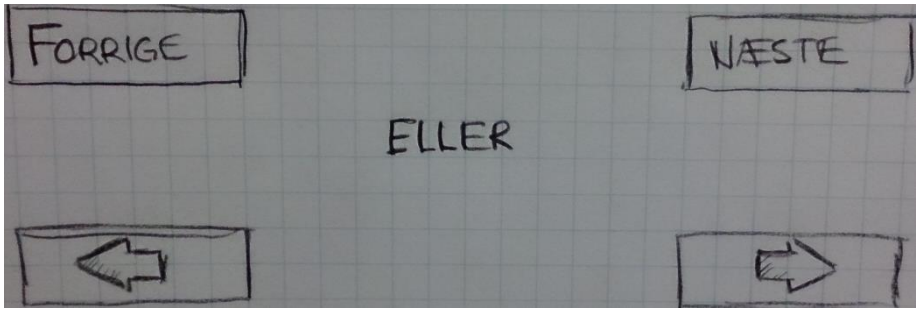
1. Ønskes der knapper til at skifte imellem pictogrammer i en sekvens?
2. Ønskes der en scroll-funktion til at skifte imellem pictogrammer i en sekvens?

3. Ønskes der en kombination af mulighed 1 og 2?
4. (Ønskes der noget helt andet?

***Valg 6** – Hvordan skal udseendet være på en evt. knap til at skifte frem og tilbage imellem pictogrammer?*

På BILLEDE 6, er der vist 2 forslag til udseende af eventuelle knapper. Disse knapper ville være placeret i de firkanter man kan se i bunden af BILLEDE 5.

1. Tekst med ”Forrige” og ”Næste”.
2. Pile
3. (Noget helt andet?)



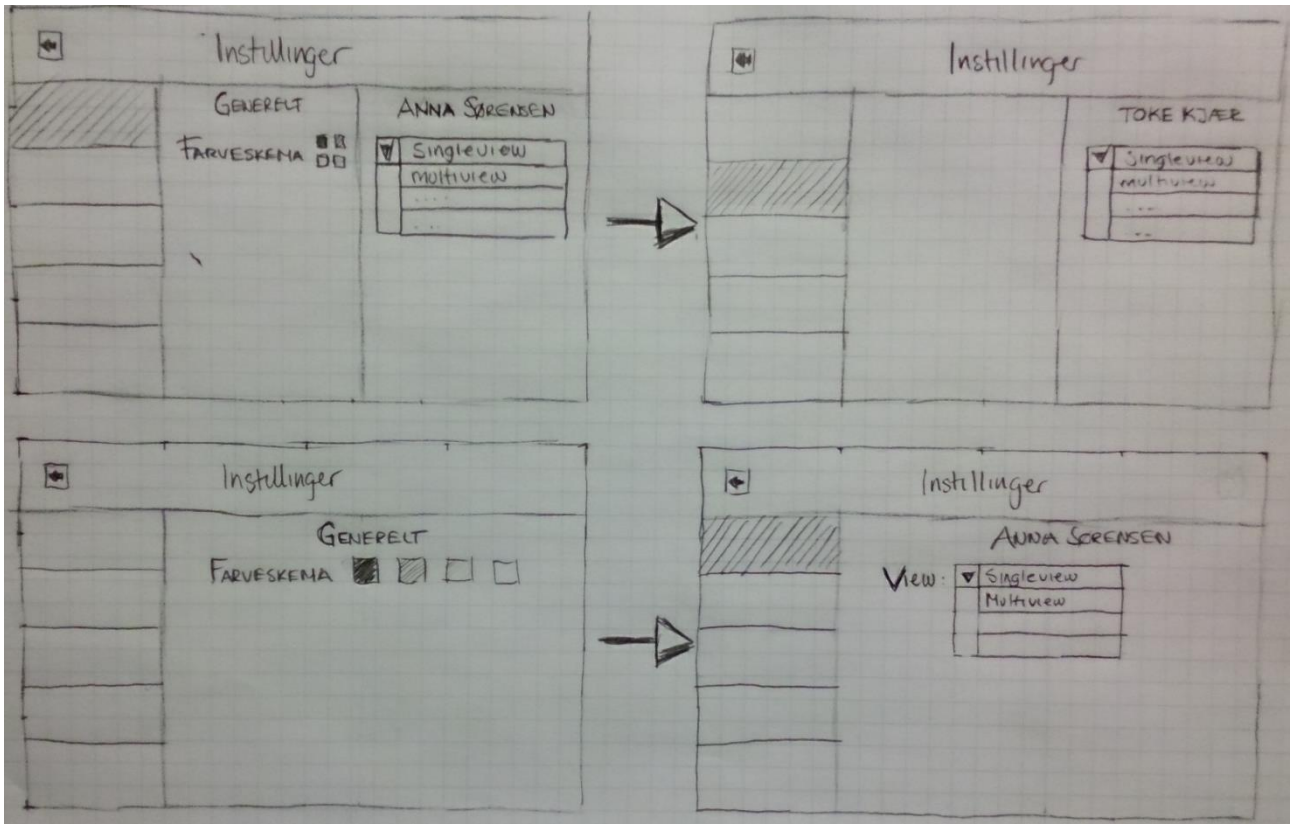
BILLEDE 6 – På billedet er der vist 2 forslag til hvordan eventuelle knapper til at skifte imellem pictogrammer i en sekvens kan se ud.

***Valg 7** – Ønskes der mulighed for indstillinger?*

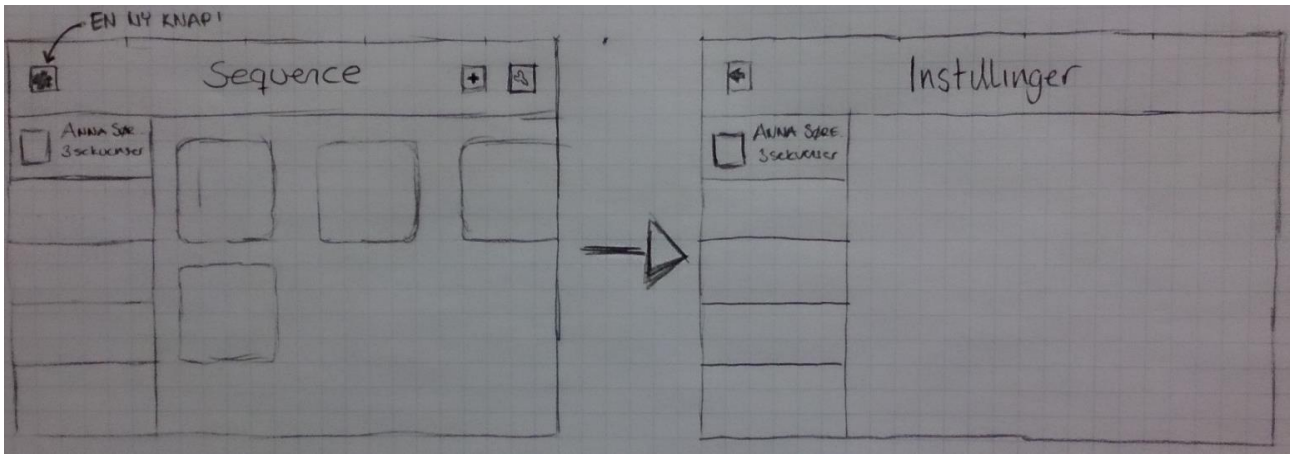
Hvis stakeholder gerne vil kunne præsentere sekvenser i single-view og multi-view, afhængigt af hvilket barn tabletten gives til, hvordan skal det så være muligt at skifte de indstillinger.

1. Ønsker stakeholder mulighed for at lave indstillinger til hver enkelt barn?
2. Ønsker stakeholder ikke mulighed for at lave indstillinger, hvordan ønskes der så mulighed for at ændre single-view vs. multi-view? (HVIS STAKEHOLDER VALGTE OPTION 3 I VALG 3.)
3. (Ønsker stakeholder noget helt andet?)

Tag gerne en diskussion om dette ud fra BILLEDE 7 og BILLEDE 8. Tag også gerne en diskussion om hvilke indstillinger stakeholder kunne tænkte sig i Sequence-applikationen.



BILLEDE 7 – Et udkast til hvordan man kan lave indstillinger for multi-view og single-view. De generelle indstillinger med farveskema er i dette billede kun med som del af skitsen, ikke nogle der direkte skal være i Sequence.



BILLEDE 8 – Et andet udkast til indstillinger, hvor der er lavet en ny knap øverst til venstre. Her vil man gå ind i indstillinger for valgte barn og her kunne man så indsætte de indstillingsmuligheder stakeholder ønsker.

Appendix C

Result from customer meeting

The following document is the feedback received from the requirements meeting, using the requirements report from B.

Requirements

Skolen
Slør billeder der ikke er aktuelle i fokus
Lade markeringen/pilen være stationær, og flyt med billederne
Hvor mange billeder der skal vises: fleksibelt
sviping er bedre end knapper til at vise næste/forrige pictogram
* Det skal være muligt at låse en aktivitet for børnene
Et lille billede er bedre end et halvt preview

Bosted
Sekvenser skal køre oppe fra og ned!!
Evt. eget valg af markør? de bruger pilen da det hører under teorien THEACCH. Den er kendt
Bedre at flytte markøren end billederne
Vis hele sequencen på samme tid. Mulighed for at tilføje flere billeder til allerede lavet sekvens.
* Mulighed for at låse aktiviteter.
Minimum to billeder af gangen da de skulle kunne se næste aktivitet.
6-7 billeder er max længde
Et overstået billede skal helst forsvinde, ellers distrahere det.
Ingen halve billeder!
sviping fungere
standard billeder er 3x3
Hvorfor spørger den om gem hvis der ingen ændringer er?

Appendix D

Requirements specification

The following is the initial requirements report created solely by the Requirements group. Only the item "Sequences (Zebra)" is relevant for Sekvens.

1 Specification Requirements

1.1 Functional Requirements

- General
  - The solutions should implement user profiles, enabling easy switching between users.
  - Split the app into administration profiles and user profiles.
  - Lock the administration for user profiles.
  - The solution should have shared pictures in every subprogram.
  - The solutions should run without any crashes and features should work as intended.
  - The user profiles should save the customized settings and categories of pictures.
- Categorizer (Cat)
  - The guardians can categorize the pictures.
    - Each user profile have customized categories.
    - Guardians connect specific pictures to the users categories.
  - The guardians should be able to easily select between a presented selection of categories.
  - Options for creating new categories, edit existing categories or remove a category.
  - A list of user profiles which the guardians can connect categories to.
- Sequences (Zebra)
  - Guardians can make predefined sequences for later use.
  - The user are locked from editing the sequence.
  - The user should be able to mark how far in a sequence they are.
- Life Stories / Week Scheduler (Tortoise)
  - An option for the guardians to set the preferences for the given user profile regarding the number of displayed pictures.
  - The solution should provide a tool for planing daily activities.

    * The planing tool should be capable of showing different amounts of activities at a time.
    * The solution should enable the possibility for the user to have a choice between predefined activities.
    * The user should be able to mark how far in a sequence they are.

- PictoCreator (Croc)
  - Guardians can easily create pictures
  - Guardians can edit pictures.
  - Guardians can add texts to pictures.
  - Guardians can record the pronunciation of the meaning of the pictures.
- Timer (wombat)
  - The solution should digitize the analog timers.
- Communication Tool (Parrot)
  - Users can play the recorded sound of a picture.
  - It should be possible for the user to make sequences of pictures for communication.
- New Tools
  - A week- and day schedule for Birken's 55" screen.

1.2 User Interface Requirements

- General
  - The color scheme should be black and white.
    * The guardians should be able to change the color scheme of specific programs and users.
  - The solution should react and give feedback on each input.
  - It should be indicated whether or not a picture has sound attached.
- Sequences (Zebra)
  - There should be different ways to view the sequences, such as show three pictures at once or just one picture.
  - A method for users to move through the pictures in a sequence.
- Life Story / Week Scheduler (Tortoise)
  - The week schedule should mark each day with specific colors.
  - The week schedule should have three different views, either a week at a time, a day at a time or a single task at a time.
- Timer (Wombat)
  - The buttons in the timer should be able to be vertical or horizontal aligned.

  - Different visual representation of how time progress for the selected amount of time.

- New Tools
  - Week scheduler should have different colors for each day.

Appendix E

Database Schema for sequences

The following is the agreed upon database schema, designed in the collaboration between the LivsHistorier, PiktoOplaeser and iOS groups (See 6.1). The variable `posY` is not used by Sekvens.

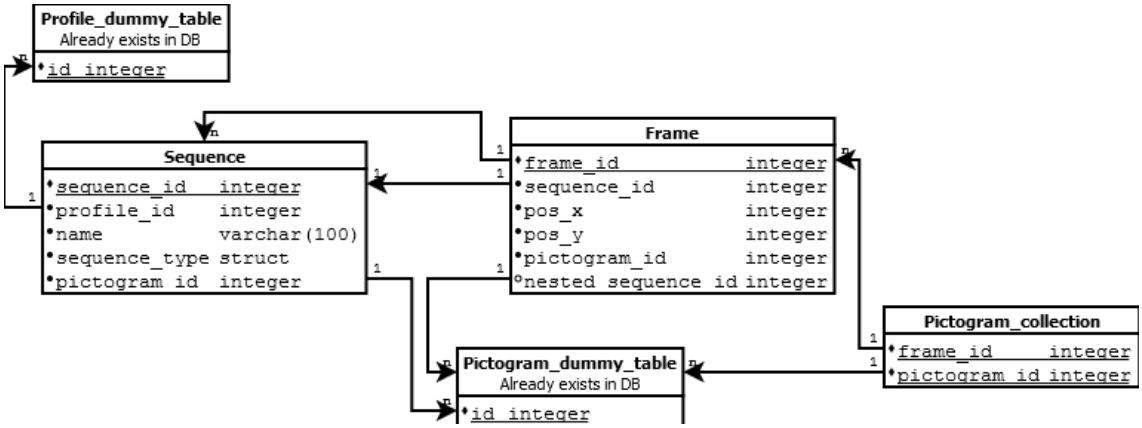


Figure E.1: ER diagram of sequence schema

Appendix F

Assignments for usability test

The following is the list of assignments used to make test subjects navigate Sekvens during the usability test. It should be noted that names and pictograms varied between the test subjects. The following list of assignments displays the names and pictograms used in the first usability test.

IDA – Opgaver til usability test

Information før test

Værgen er Tony Stark som har en liste af børn. (Vis i Launcher)
Til testen er der kun pictogrammer bestående af forskellige farver.

Opgaver

1: Lav en ny sekvens med navnet "Vask hænder" til Lone med 3 gule pictogrammer og gem.
2: Lav en ny sekvens med navnet "Gå på toilettet" til Lone med 2 røde pictogrammer. - Indsæt sekvensen "Vask hænder" mellem de 2 pictogrammer og gem den.
3: Tilføj et valg bestående af et gult og et orange pictogram til "tag sommertøj på". Valget skal være det første i sekvensen - Vis sekvensen. Vælg den orange farve i valget. Gå derefter tilbage og gem sekvensen.
4: Åben "Spis morgenmad" sekvensen. Vælg et nyt billede til sekvensen. - Skift det tredje pictogram til et rødt pictogram. Slet det første pictogram. - Vælg derefter, ikke at gemme din ændring.
5: Kopier "Spis morgenmad", "Tag sommertøj på" og "Gå på toilettet" til William og Magnus.
6: Slet sekvensen "Spis morgenmad" og "Gå på toilettet" fra Magnus.
7: William foretrækker at se 3 pictogrammer af gangen. Skift denne indstilling.
8: Luk Sekvens.

Johan åbner Sekvens med William som borger.

9: Vis Williams sekvens "Tag sommertøj på". Scroll til enden.

Appendix G

Resume of report

For the last few years, Software 6 studens have worked on a series of applications under the common project name GIRAF. These applications are designed to help individuals suffering from autistic spectrum disorder and the ones who help take care of them.

Because persons suffering from autistic spectrum disorder often have significant disabilities, being a caretaker is not always an easy task. Common disabilities are the lack of ability to communicate. Some will for example have a very small vocabulary if any at all. This is an obvious hindrance both for the affected persons but also the people they interact with.

GIRAF puts focus on some of the difficulties and attempts to provide tools to help overcome them. One of the applications within GIRAF is Sekvens. This application is designed for caretakers to create sequences of pictograms in order to help guide a person with an autism disability through otherwise difficult tasks. Every person could then have their own unique list of sequences, created by their caretaker(s).

Depending on the degree of handicap, a sequence could be as simple as a series of pictograms showing the steps required to wash hands properly.

With Sekvens being an existing application from previous Software 6 semesters, a codebase existed at the beginning of our semester. It was graphically able to create and edit sequences with no additional features. It was however also very close to be a stand-alone application as it was not tied into GIRAF. It was for example not set up to cooperate with other GIRAF projects such as the database and the Launcher which is where applications were meant to be launched from.

For this semester, the goal has been to finish Sekvens and tie it into GIRAF the way it was meant to. This was done, using the feedback of customers as guidelines. Throughout the semester Sekvens had its codebase reduced, although preserving functionality. This was a consequence of tying it into other projects, depending on others code. A lot of new features were designed on requests of customers. These includes management tools to for example copy M sequences to N persons. Sequences themselves has also been advanced to include new features such as letting the user create choices for the person suffering from an autism spectrum disorder.

Sekvens was taken to a near-complete state where almost all customer requests had been addressed. The report covers this process in detail.

At the same time, a second project was started half-way into the semester. A few other projects were also supposed to display sequence-like series of pictograms. For this reason development was initiated on a Sequenceviewer which ideally should be able to play sequences for anyone wanting to use it, while also customized for everyone, depending on their needs.

This project was however not finished. The report covers the Sequenceviewer from the initiation to the final state for this semester - a viewer with the core functionalities implemented, although still lacking a few requested features.