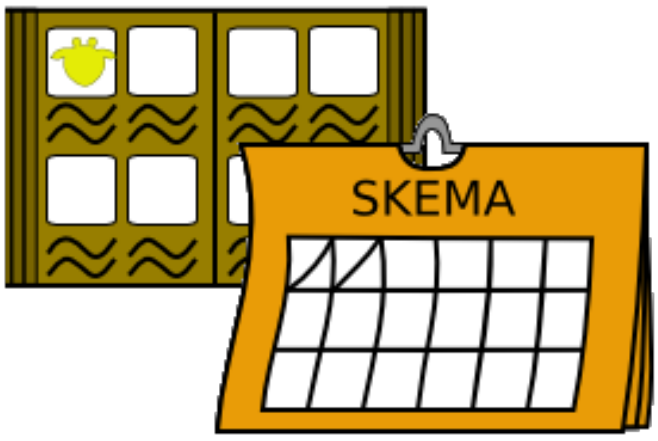
Week Schedule and Life Stories

Group sw602f14



Spring 2014

AALBORG UNIVERSITY
STUDENT REPORT

Department of Computer Science
Software 6th semester
Address: Selma Lagerlöfs Vej 300
9220 Aalborg Øst
Phone no.: 99 40 99 40
Fax no.: 99 40 97 98
Homepage: http://www.cs.aau.dk

Project title:
Week Schedule & Life Stories

Subject:
BSc Project (Developing Complex Software Systems)

Project periode:
Spring 2014

Group name:
sw602f14

Supervisor:
Xike Xie

Group members:
David Bachmann Jeppesen
Christoffer Ndũrũ
Dan Skøtt Petersen
Kristian Mikkel Thomsen

Copies: 6

Pages: 56

Appendices: 4

Finished: May 28, 2014

Abstract:

The purpose of this project was to develop two Android apps for people with an Autism Spectrum Disorder, henceforth known as citizens, and their guardians. The two apps were developed in a multi-project environment with dependencies to other apps. Some of the citizens have little or no language, which creates the need for images (pictograms) when communicating with them. The aim of the apps was to replace tools already in use in the institutions, relieving the guardians of difficulties in managing the vast amount of physical pictograms. The Week Schedule app should replace a bulletin board used to schedule daily and weekly activities for the citizens. The Life Stories app should replace a book the citizens use for telling and remembering their experiences. Both apps were completed and functional and were deployed on the customers' tablets at the final sprint end meeting.

Preface

This report was written by four 6th semester software engineering students from Aalborg University.

The report details the work the group did to produce two Android apps (Life Stories and Week Schedule) for use in the Graphical Interface Resources for Autistic Folk (GIRAF) framework. The language of the apps is Danish by request of the customers, however, throughout this report they are referenced by their English names. Beyond describing the work done by the group, the report also describes the work performed in collaboration with other groups.

The source code for this project is accessible through one of the following links:

- `https://www.dropbox.com/s/od3vdobckuhtq9a/tortoise.zip`
- `http://goo.gl/CHRxEp`

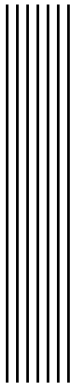
We would like to extend thanks to our supervisor, Xike Xie, for providing guidance throughout the project, and to the involved customers from the institutions for their feedback and ideas. We would also like to thank Ulrik Nyman for starting the GIRAF project.

David Bachmann Jeppesen

Christoffer Ndũrũ

Dan Skøtt Petersen

Kristian Mikkel Thomsen



Contents

1 Introduction

People diagnosed with an autism spectrum disorder (ASD) can suffer from a wide variety of mental conditions, such as language difficulties, problems with social interaction, and a tendency to conduct limited repetitive actions[1]. This can make everyday life difficult for people with ASD and their surroundings.

Citizens with severe degrees of ASD are enrolled in specialized institutions in order to meet the special needs of their individual inabilities. These institutions use tools to provide an optimal learning environment for the citizens, based on their respective stage of development. Many of these tools utilize pictograms when communicating with the citizens, as many of them have not yet developed a spoken language. Figure 1.1 shows how pictograms are organized at an institution.

Figure 1.1: Pictogram archive at an institution.

A large amount of pictograms are needed, as some citizens lack the ability to abstract classes of objects from specific attributes of the object. One example of this, provided by one of the customers at the first customer meeting, is that if a citizen with ASD has identified a yellow bus as a bus, that citizens might not be able to

recognize a red bus as a bus. This inability to classify objects creates a need for a large amount of pictograms to ensure good communication between guardians and citizens. Managing lots of physical pictograms is difficult and takes up a lot of time for the guardians. Time that is better spent on the citizens.

1.1 Week Schedule

Most people with severe ASD have a common sensitivity: they become insecure when facing new or unexpected situations. This can result in strong destructive reactions from the person with ASD, which is undesirable. To accommodate this sensitivity, every citizen has his/her own schedule for the week, such that the citizen always knows what the next activity will be.

In reality, this week schedule is currently a bulletin board with activities in the form of laminated images and pictograms. Every week, an institutional guardian must clear this bulletin board and set up the entire week using the laminated images. A lot of time is wasted on clearing the board and finding the correct images and pictograms. One of the main objectives of this project is to make this procedure easier for the institutional guardians by developing an app, that can be used to create and display week schedules. When using this app, the institutional guardians should be able to clear the board by the push of a button, save templates, so that frequently used outlines of a schedule can be easily loaded and save complete schedules.

1.2 Life Stories

Another tool frequently used in the institutions are small books designed to create sequences of pictograms in them. These books are utilized to help citizens who have no spoken language to tell stories about what they experience, and maintain memories of these experiences. This helps them develop their personality which, according to one of the customers, can be an issue for people who have ASD.

Using a physical book with physical pictograms has some drawbacks. When having many pictograms to choose between, it becomes difficult to find the right ones quickly, and creation of the stories becomes a factor of frustration for the institutional guardians.

The other main objective of this project is to finish an app to handle creation and display of these Life Stories. The work on this app has been ongoing from previous 6th semester students and the aim of this project was to complete the app and make it useful in the institutions. Having the pictograms categorized on a tablet and the app available to organize them in life stories, makes the task of creating the stories easier.

1.3 Roadmap

The following is a short description of each chapter in the report.

Project Management
The role of this project in the context of the overall multi-project and a short description of how the contact with customers was handled.

Sprints
High level documentation of what have been done in each sprint.

Collaboration
Detailed descriptions of the collaboration between this project and some of the other projects.

Design and Implementation
Contains descriptions of certain places in the project code where some important design choices were made.

Test
Documentation of the testing performed on the products.

Future Work
Contains lists of bugs, refactoring issues and future development areas.

Conclusion
A recap of the project work and an overall conclusion for the project.

Reflection on the Multi-project
Some thoughts about the nature of working in a multi-project and future work on the product.

2 Project Management

This chapter describes how the whole multi-project was organized, how customer collaboration was secured, and how requirements were managed.

2.1 Multi-project Environment

The main purpose for the semester was to develop a complex software system in a larger development environment. The overall project is called GIRAF (Graphical Interface Resources for Autistic Folk), and it consists of a number of individual sub-projects, each with their own purpose.

The following is a list of all the sub-projects:

- Life Stories and Week Schedule (Tortoise): The two apps this report is focused on.
- GIRAF, referred to as *Launcher* in this report: The launcher app responsible synchronizing local-db and remote-db, launching other apps and managing settings of the apps.
- Local-db: The database on the tablets. Part of launcher as of sprint 4. Oasis-lib is used to access data from local-db.
- Remote-db: The database hosted on a separate server. Contains data not generated by the apps. Mananged by Giraf Admin.
- Giraf Admin: The web based administration tool for the GIRAF. Responsible for creating users and associated QR-codes.
- Oasis-lib: The library used to read and write to Local-db.
- Oasis-App: A lite tablet version of Giraf Admin able to manage citizens associated with a currently logged in guardian.

- Timer: An app able to display a time both as an overlay on the other apps and a full screen timer.
- Sequence: An app for creating sequences used for display daily procedures for citizens.
- Sequence Viewer: Common sequence viewer, used by Life Stories and Sequence
- Picto Painter: An app for creating pictograms.
- Category Tool: An app for managing pictogram categories.
- Visual Scheduler (iOS): An iOS standalone version of Week Schedule.
- PictoSearch: The app used to search and retrieve pictograms from Local-db.
- Pictogram to Speech: An app for helping citizen communicate by converting pictogram to speech.
- iOS Speak for Me: A iOS standalone of Pictogram to Speech.
- Giraf Components: The library for shared GUI elements.
- Category Game: A game for training the associative abilities of citizens.
- Voice Game: A game for training citizen in maintain a some adjustable voice levels.

This project focuses on one aspect of GIRAF codenamed Tortoise, which covers the development of the existing Life Stories and the new Week Schedule app. Figure 2.1 shows the other projects Tortoise depends on to function correctly.

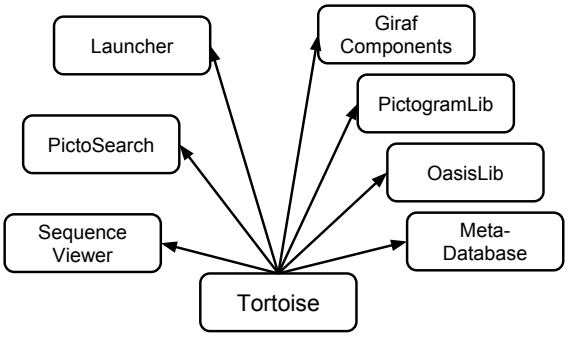


Figure 2.1: Figure showing the dependency relations of this project in relation to the other projects.

To manage GIRAF, several responsibilities were distributed between the groups. These responsibilities cover management of Redmine (issue-tracker), Git, Jenkins (continuous integration), host server, Android, Google Analytics, and creation of app icons. Additionally, people were appointed to manage Scrum of Scrums, the project backlog, sprint ends, and customer contact.

Many small areas of responsibility in the multi-project had the advantage of distributing workload, such that the workload of each group was reduced. This had

the disadvantage, that some informal areas was not covered and thus was in danger of being neglected.

Having only one person to manage customer contact ensured the customers would not be overwhelmed by questions from different people. This also made it harder for each group to get quick answers for their questions, possibly resulting in them making decisions on behalf of the customers.

As the mentioned tasks indicate, a number of technologies were used to organize the project. The overall project organization was done using Scrum, and each sprint lasted for about a calendar month. The earlier sprints were longer than the later sprints, so as to ensure an equal amount of time to work in each sprint, due to more lectures earlier in the semester. To ensure frequent communication between the different groups, a Scrum meeting was held every week with the Scrum Master of each group. And each sprint had a sprint start and end meeting, which was for everybody involved in the multi-project. Additionally, it was agreed that each group should use Redmine to track their development issues, so there would always be an overview of the progress of the entire project.

The need for managing the whole project resulted in extra workload, and instead of being able to focus exclusively on our own development process, we had to take Scrum meetings, sprint meetings, and meetings with other groups into account. The workload mainly consisted of coordinating with other groups if we needed a feature e.g. when we needed a modified database scheme. The database group modified the scheme and we had to test and use it. Afterwards we showed the Zebra group how to use it. As a result the sequence viewer was compatible with Life Stories. In spite of the extra workload it became clear that it was necessary to communicate with the other groups to stay on track and it could have the benefit of saving us time. If we did not coordinate with the other groups, we could run into trouble. We ran into trouble when the PictoSearch group was hard to reach and they committed a new release of their app. This broke our app until we fixed it ourselves. The trouble occurred because we were unable to notify them of our problems.

The next section will be describing how a tool called Redmine was used in the context of the multi-project.

2.2 Redmine Specialist Responsibility

A lot of the project management regarding the multi-project was done using a tool called Redmine. Redmine is a free, open source, web-based project management and bug-tracking tool. It implements inheritance-functionality in both projects and issues and every sub-project has its own corner with optional forum, wiki and issue-tracker. Two specialists was assigned to administer Redmine - one of them being a member of this group.

In the beginning of the project-period there was some work on Redmine in creating new projects, groups and assigning students to these groups. The following Redmine assignments was carried through by the specialist in this group.

In collaboration with the Issue-tracker specialist from group sw603f14, the trackers Feature, Report and Refactoring was added. Already existing trackers was also evaluated, but because they were already used in issues, it was decided not to remove any of them.

A new status called iced was introduced in order to be able to mark issues as on hold.

A guide for creating local versions on Redmine was created. All software issues should be assigned to a version defined on the multi-project level and in order to keep track of the overall progress of the project. However, some groups also wanted to track their report progress using Redmine issues. As this is irrelevant for the multi-project, these issues should be assigned to one or more local versions in their own corner of Redmine.

Redmine was the main tool for managing the projects - the next section will go on describing how contact with the customers was handled.

2.3 Contact with Customers and Requirements

Periodic contact with the customers for whom the product was made, was important in order to ensure that the apps were up to date with the needs of the customers. Contact between the group and customers was not handled directly, as all communication with the customers was achieved through a contact coordinator. The other task of the coordinator was to set up meetings, such that if a design change was proposed by a group, it was possible to ask the coordinator to schedule a meeting with the customers, in order to get feedback.

The customers might express the need for a change in the existing apps or the addition of a completely new feature. In this case, the group would discuss the new request and decide whether it was feasible within the time frame available for the apps. If this was the case, the requirement was decomposed into smaller tasks and they were added to the next sprint back log.

At the end of the project a user test was conducted. In the test a customer was required to go through several tasks in the two apps. While they were performing the tasks, the difficulties they had were noted and rated according to their severity. A more detailed account of the user test can be found in Section 6.2.

3 Sprints

The project was split into four sprints, each consisting of talking with the customers as well as designing, implementing, and testing features for the app. Some aspects of the app were worked on during more than one sprint, and therefore the implementation is described in Chapter 5 while progress and reasoning are described in this chapter.

3.1 Roadmap

The primary focus for each of the four sprints was as follows:

Sprint 1
Understanding and commenting the existing code.

Sprint 2
Refactoring, GUI improvements on Life Stories, start of the new Week Schedule app, and collaboration with other groups.

Sprint 3
Development of the Week Schedule, big changes to the Life Stories GUI, and continued collaboration with other groups.

Sprint 4
Finishing and polishing both apps.

3.2 Sprint 1

In the first sprint the main focus was on understanding the existing code written last year, as well as on correcting the most critical issues still present the app. Other than minor tweaks, no considerable effort was put into developing new functionality. The planned activities and their execution are described in this chapter.

3.2.1 Goals

For the first sprint there were a number of initial goals. The goals for the first sprint were as follows

- Understand and comment the code.
- Fix bugs encountered in the code.
- Make it more intuitive to change profiles between citizens.
- Make an exit button to make it more clear how to exit the app.
- Improve the highlighting of pictogram frames.

The individual goals are described in more detail in following sections.

3.2.2 Understanding, Commenting, and Refactoring

The development of the Life Stories app was started last year, and the focus has clearly been on making a working app rather than writing good code. As a consequence the code was poorly commented, the methods are long – some times several hundred lines, and there was a lot of duplicate code.

The first thing that had to be done before moving on to the next task was understanding the code. This was, partly due to the lack of comments, a big task.

After understanding the code some of the missing comments were written. The code standards for the semester prescribed that a Javadoc comment was to be made for each class, and to make the comments more organized, we agreed to also use Javadoc comments for methods.

Commenting all the code was a big task, and making it the main priority would have resulted in very little other work being done. Therefore it was decided to make it an ongoing task. At the end of the sprint, approximately one third of the code had comments.

Only little refactoring was made during this sprint, but several notes were made about which parts of the code needed refactoring. The two most important areas focused on, were bringing down the size of certain methods and reducing the amount of duplicate code.

3.2.3 Usabilliy Issues from Last Year

This section contains a description of usability issues from last year's work along with the solutions implemented to fix them.

Profile change

In the report from last year, it was mentioned that the way one should change between citizens' profiles was confusing. The app from last year had a profile picture in the upper left corner. To the users it seemed intuitive to change profiles by clicking

the profile image. We implemented the feature in this sprint, in collaboration with the group responsible for the launcher, sw605f14. We had to collaborate with them, because the profile selector screen which our app goes to when the profile picture was clicked, is located in the launcher.

The collaboration with the Launcher group consisted mainly in coordinating how our app started the profile selector activity in their app. At first it was not expected to take up a lot of time to implement this feature, but it turned out that several issues arose during the implementation of the feature. It took a considerable amount of time to find out exactly what needed to be changed, and it turned out to mainly be a problem with how the launcher app handled intents.

The first several fixes of the launcher proved unsuccessful and a lot of debugging was needed. Eventually the profile change was implemented sucessfully.

Access to the profile selector screen in the launcher was deemed a useful feature because some of the other groups also wanted to be able to use the launcher's profile selector.

Exit-button

Users did not find it intuitive to use the hardware buttons and so, a button was implemented, that closes the app and returns to the previous screen, which is the launcher. This button was actually implemented in a slightly different way than most of the other buttons in the app. The other buttons get their listeners set in the MainActivity class, but this button has an `onClick`-attribute in the XML ressource-file, which invokes a method.

Frame Highlighting

In the usability test from 2013 one of the usability issues was that the selected pictogram frame was difficult to distinguish from the other frames while in guardian (edit) mode. Previously this distinction was made by enlarging the selected frame and giving it a lighter border. As the frames can vary in size, the focus was on making the border stand out more. This was achieved by adding a dashed border to the selected pictogram frame.

3.2.4 Summary

The primary focus of sprint 1 was to get familiar with the existing code which was partially achieved through commenting the code. All goals were reached to some degree: a third of the code was commented, all usability issues from last year were addressed, and a crash when attempting to render a pictogram with no associated bitmap was resolved.

3.3 Sprint 2

In the end of the first sprint sw615f14, the group responsible for the database API, announced they would be releasing a new version of the database API in the beginning of the second sprint. As a consequence all apps had to be modified to

accommodate these changes. It should be noted that some requirements changed during the sprint, causing some of the goals stated below to become less relevant. Furthermore, some goals were worked on in collaboration with other groups.

3.3.1 Goals

For the second sprint the goals were as follows:

- Analysis and design:
  - Collaborate with other groups to design new database and GUI.
  - Create prototypes for the Week Planner app, and verify requirements with customers.
- GUI refactoring:
  - Use the new, shared, GUI elements (buttons and dialogs).
  - Improve buttons for adding content to frames.
  - Make it possible for the user to choose a background color.
- Code refactoring:
  - Make Tortoise work with the new database design.
  - Divide project code into packages.

The details and progress of each goal are described in the following sections. Chapter 4 describes the collaboration done with other groups.

3.3.2 Adapting Life Stories to the New Database API

The 2013 GIRAF semester left behind a local and a remote database with incompatible schemas. The group assigned to the local database, sw615f14, changed the schema to be like the remote database during sprint 1. This caused problems for all Android apps in the multi-project. At this point Life Stories used the local database to get profile and pictogram data, but not for saving and loading sequences. As mentioned, Tortoisedepended on the apps shown in Figure 2.1, and in addition to adapting the code interfacing with the local database API, all the dependencies of Life Stories had to do the same in order for Tortoise to function.

As a result of this, Life Stories became independent of the launcher for most of the second sprint, and our group developed a modified version of PictoSearch to prevent it from crashing during the second sprint review.

3.3.3 Using the Shared GUI

In this sprint sw603f14, the group responsible for the shared GUI elements, announced that they were introducting a new GUI with the purpose of making the buttons and dialogs uniform across all apps. As a consequence the Life Stories app also had to use the new GUI. There were some initial difficulties replacing the old

code for buttons and dialogs which took up some time, but afterwards it was relatively straight forward. There were also further modifications made by sw603f14 after the new GUI's initial announcement, which required some modifications to Life Stories.

3.3.4 Better Buttons for Adding Content

Initially in this sprint, the group wanted to improve the buttons for adding pictograms in various contexts in the app. This turned out not to be necessary because of the planned creation of a partially shared GUI with the Zebra group. Thus, no time was spent working on this sprint backlog item.

3.3.5 Evaluation of Sprint 2

Adapting to the new database was more time consuming than expected. In addition to work on Life Stories, time was also spent working on the dependency PictoSearch. An addition to the new database schema was designed in collaboration with other groups with similar data structures, to include database tables for sequences. Dividing code into packages and using the new shared GUI elements was only done partially. We collaborated with group sw610f14, who were making the Visual Scheduler app for iOS, about the intial design of Week Schedule.

3.4 Sprint 3

In this sprint the time required to follow the courses started to decrease, and enough time had been spent on getting an understanding of the existing code that development of new or large features could begin. This was the primary focus in this sprint.

3.4.1 Goals

- Week Schedule:
  - Create a working, but unpolished, app according to the specifications (see Section 5.4 and Appendix A). It should be ready to be presented to the customers at the third sprint end.
- Life Stories:
  - Move the right menu bar to the top.
  - Move the options for adding content to a frame from the menu, to a dialog box.
  - Allow the user to set a custom icon for choice pictograms (when multiple pictograms were chosen by the guardian for one frame).
  - Change the way frames are added to be in straight lines rather than dragging and dropping in a big area.
- Collaboration:

- Use OasisLib to save sequences.
- Design a sequence viewer.
- Design the Week Schedule app.

3.4.2 Week Schedule

In this sprint, one of the major goals was to create a week scheduler for the customers. Its primary purpose was to allow guardians to create weekly plans for the citizens, giving both citizens and guardians an overview of when they are going to do which activity.

The guardian (edit) mode was created in this sprint, although certain features were still unpolished, like being able to save a week schedule was missing. The citizen (view) mode was still incomplete, but it was expected that much of the code from the guardian mode could be reused for implementing the mode.

In guardian mode, it was possible to add frames to a chosen week day as well as remove them. It is also possible to go into portrait mode, although this feature was yet unfinished. This resulted in frames disappearing, when changing from landscape to portrait mode and vice versa. It is also possible to add more frames to a particular day, than a device can show at one time, by using scroll views.

3.4.3 Life Stories

Some big changes were introduced regarding the look and feel of the Life Stories app. The idea behind these changes were to give the user a more streamlined and intuitive experience when using the different apps. It was decided to remove the menu appearing on the right side when editing a life story, and instead introduce a menu in the top, as was already the case in the rest of the app. Additionally, a dialog box was added to edit the individual frames, rather than showing it on the menu bar as before. This dialog box included an option for the user to choose a custom icon for choice frames (frames with more than one pictogram associated).

The option to add frames was moved from the menu panel, to the actual view where the frames are shown, removing the confusion of having to drag a frame out to its desired position, as well as the ability to put frames in completely random positions. This functionality was implemented based on the code made by the group that was responsible for the Sequence app last year, and even though the option to drag and drop anywhere was removed, it is still possible to drag and drop the added frames to change their position. This new way of adding frames removes the option to place a frame anywhere, and only allows them to be in a sigle line.

During the remake of the layout, two functionalities were deemed more confusing than useful, and therefore removed entirely. These were the option to switch between other saved stories, when in citizen mode, by using arrow buttons, and the option of grouping frames, such that they contain the same pictogram or pictogram-choice.

Overall, these changes resulted in a more streamlined app, with fewer ways to do something unintended, which is greatly desired, as the citizens can have a hard time adapting to random changes. While the app seems more polished after these changes, it also resulted in it having almost the same functionality as the Sequences app. Consequently, a decision was made to reduce the effort spent on developing it further, and instead focus on the Week Schedule app, which the customers expressed a much greater interest in and need of.

3.4.4 Saving Sequences Using OasisLib

One of the big open issues from the previous work on Tortoise was that it did not save sequences using OasisLib. The design of the database prevented this, as there were no tables to save the data in. These tables were designed during sprint 2 (see Section 4.2.2) and the group responsible for implementing these tables and updating the API, sw614f14, did so during sprint 3.

The sequence class used in the database API differed slightly from the sequence class in Tortoise. Extra requirements from other projects and specialized functionality in the Tortoise sequence made it difficult to use the same class directly when working with the database API. The solution for this was to implement a translator for converting between the two classes. This translator was called DBController, as all communication with the database regarding sequences is handled by this class.

3.4.5 Evaluation of Sprint 3

The Week Schedule was made, but was unpolished. It was designed in sprint 2 in collaboration with the sw610f14 (the Visual Scheduler group). A semi-working portrait mode was also implemented, but when orientation was changed, pictograms were not transferred to the new screen view. To make sure we would be able to deliver a working product, work on the portrait mode was stopped after this sprint. Guardian mode was implemented, but it was still not possible to save schedules, and citizen mode had not yet been implemented.

A sequence viewer was designed in a collaboration with sw607f14 (the Sequence group).

For Life Stories, the menu bar was moved to the top and the options for adding content to a frame was moved from the menu to a dialog. It was also made possible for the user to set a custom icon for choice pictograms (multiple-pictogram frames).

The option to place pictograms (frames) anywhere on the screen was removed, now only allowing them to be placed in a straight line. Drag and dropping of pictograms was still possible, but only on one line and not the same big area as before.

The new database API was incorporated into the code, so it was possible to implement functionality to save schedules and life stories in the next sprint.

3.5 Sprint 4

This sprint was the final sprint of the project. By the time this sprint was completed, no further work could be done on the app, and therefore the goals for this sprint was focused on finishing ongoing work, implementing important missing features, fixing bugs, and polishing the GUIs. Near the final deadline the remote database was connected, and the app used to select pictograms from the database, PictoSearch, stopped working. The group responsible for this app was not available, and some effort was put into debugging it.

3.5.1 Goals

- Week Schedule:
  - Implement a way to make choices, as in Life Stories.
  - It should be possible to save a schedule and edit an existing schedule.
  - Implement a citizen mode.
    - Mark the current day and activity.
    - Scrolling in citizen mode should only allow full pictograms to be shown.
    - There should be a way to cancel an activity without deleting it.
- Life Stories:
  - Implement the feature of sending a sequence to a specified email address.
  - Make it possible to edit existing life stories.
- Overall:
  - Install both apps on customer tablets at the sprint end meeting.

3.5.2 Week Schedule

In this sprint, the Week Schedule app was modified so it could also handle frames with several pictograms asosciated. This took some time as several problems were encountered, which were not encountered during implementation with only single pictograms.

A citizen mode was implemented in which the citizen could only view pictograms and select single pictograms from choice pictograms (frames with multiple pictograms). It was easier to implement than guardian mode, since the citizen mode was mostly the same as guardian mode with some of the buttons and functionality hidden.

The rough version of Week Schedule from last sprint was using standard Android ImageViews to show pictograms in their respective weekday. This was changed to use our own media frames for displaying pictograms.

Another thing we had had a lot of trouble with was the database. In this sprint pictograms could finally be loaded from the shared remote database containing thousands of pictograms, which turned out to be too much to handle for the PictoSearch app. This work required to fix this is described in Section 4.2.4.

To allow saving a week schedule, an entire week consisted of a sequence with seven nested sequences in it – one for each day. Support for deletion and saving of sequences from and to the database was implemented using the interface for the database that the database group, sw614f14, had created.

Some time was spent on converting the last buttons from the old button type to new GButtons that sw603f14 (the GUI group) had made, to give all the apps a common look and feel.

Another feature implemented in the Week Schedule app this sprint, was arrows on week days when in citizen mode. The arrows were shown when there were more pictograms on a weekday than the screen could contain (currently four). The citizen could then click these arrows to navigate up and down, having only four pictograms shown at a time, on a particular weekday. In particular, this feature ensured only whole pictograms were shown on the screen.

The customers had also requested the option for the citizen to cancel activities without deleting them. This was implemented in this sprint as well. When the a long press was performed on a frame, a red cross would appear on top of the pictogram to indicate that the activity has been cancelled. They also wanted to be able to indicate which was the ongoing activity. This was implemented so the citizen could simply touch the current activity pictogram, and a frame surrounding the activity would appear. A similar marking of the current weekday was also implemented.

The option to choose more pictograms for a single frame was also needed in the Week Schedule app. As Life Stories already had a similar feature the implementation was reused in Week Schedule with only minor alteration.

When an existing schedule is opened in edit mode it should be possible to edit and save it. Some time was spent on properly loading the sequences from the database and populating the schedule with the data. This also included updating the database controller class to overwrite the existing data in the database, when the user saves an existing schedule.

3.5.3 Life Stories

As mentioned in Section 3.4.3, not much effort was aimed at this app in this sprint. The development of new features was confined to adding the possibility of sending an image of the current life story to a specified email address, because this was a feature repeatedly requested by the customers. Other than that, focus was on fixing bugs. The feature of sending the life story to an email was derived from the customer requirement of being able to print out the life stories. With the limited time available at this point in the project, and no knowledge of the hardware and amount of technically skilled staff available in the institutions, it was deemed to be a big task to implement actual printing functionality. Therefore the easier task of

sending the life story as an image attached to an email was chosen instead, allowing the users to access the life story from a computer and print it there.

Aside from email function, only minor changes were made, including fixing the few bugs that emerged with the new functions. The option of loading and editing an existing story was also implemented, but this was mostly done the same way as in the Week Schedule app.

3.5.4 Database Changes and Synchronization

In sprint 4 database changes created extra work for our group. First in the form of an optimisation update which had errors and later on when synchronization between databases was activated.

In the middle of sprint 4 a big database update was completed by another group. This update optimized the queries used by the database library and introduced a new sub module "meta-database". The optimized queries caused some problems for some apps including Life Stories and Week Schedule as SQL error occurred when requesting sequences from the database. Once identified, the problems for the apps was quickly resolved, but as with previous database changes it takes time before all dependencies work again so the apps had to function without the Launcher and PictoSearch.

Similar problems occurred when the local and remote databases synchronized near the end of sprint 4. This caused PictoSearch to crash due to the increased data size and got our group involved in a collaboration resolving issues PictoSearch had when searching in the synchronized database.

3.5.5 Evaluation of Sprint 4

Nearly all goals were reached during this sprint. We successfully implemented choice in the Week Schedule. Marking of the current day and activity was also done. We also managed to enable scrolling on week days in a way so only whole pictograms are shown to the citizen.

Cancellation of activities as well as loading of schedules was also implemented in the Week Schedule app, but editing an existing schedule suffers from having bugs. Finally for the Week Schedule app, a citizen (view) mode was successfully implemented.

The ability to edit an existing life story was also implemented and it is now possible to send life stories as emails.

At the sprint end meeting, the apps were installed on customer tablets.

4 Collaboration

In a multi-project, collaboration between projects was essential to achieve a common goal. The fact that all involved projects, served the same purpose of making the lives of guardians, parents and citizens with ASD easier, inevitably meant that some projects would encounter similar issues that had similar solutions.

The collaboration with other groups mainly revolved around identifying common issues with common solutions. This had the advantages of reducing the workload on all groups and unifying the designs of similar apps.

The next section will establish the means of collaboration between the groups in the multi-project.

4.1 Means of Collaboration

Collaboration with the participants was formalized mainly by formal meetings and some informal talks. Exchanging general information between groups was facilitated by Redmine. The basic workings of Redmine is described in Section 2.2.

Redmine was intended to be used as a common ground for exchanging information between projects and the biggest challenge regarding this tool was the level of usage. If the majority of groups do not use it on a daily basis, the information on it becomes obsolete. The most active parts of Redmine in this project was the main project's wiki and forum, as this was where all streamlining and public discussions between groups happened.

Although Redmine offers great project management functionality, most of the communication between groups was handled directly by people visiting each other in the group rooms. Regardless of the documentational advantages of any management tool, this form of communication remained the quickest and most precise way of resolving common issues. Redmine was mainly used for conventions (e.g. for coding style), meeting summaries, contact info and keeping track of issues and progress in

these.

The following describes collaboration on specific subjects with other groups.

4.2 Focus of Collaboration

The Life Stories app is similar to the Sequence app, as both are based on creating and manipulating sequences of pictograms. It made sense to make a collaboration focused on the design of the GUI and database. The partners in the collaboration was:

- Group sw607f14 who worked on the Sequence app.
- Group sw614f14 who worked on the local database and its API OasisLib.
- Group sw610f14 who worked on the iOS Visual Scheduler app.

4.2.1 Sequence Viewer App

Collaboration tasks were to create a common viewer app for sequences and a database structure to support it.

The focus of the Sequence Viewer app was to develop a single viewer to simplify the app for the customers and save development time for groups using sequences in their Android app. Design was done by our group and sw607f14 and with feedback from sw615f14, the group working on the Piktooplæser (Pictogram to Speech) app.

Our group was appointed the task of designing the part of the database, where sequences of pictograms could be be saved. To make it compatible with all projects that would benefit from this, it was important to identify the needs of these projects. Projects using sequences were:

- Life Stories and Week Schedule by our group.
- Sequence and Sequence Viewer by group sw607f14.
- Pictogram to Speech by group sw615f14.
- Web Schedule by group sw603f14.
- Visual Scheduler (iOS) by sw610f14.

The API and implementation of database sequences were done by group sw614f14. Testing and providing requirements for the API was done by our group.

4.2.2 Database Design

As many apps handle GIRAF sequences, it made sense to create a common database design to be able to save all kinds of pictogram sequences in a cluster of tables in the database. In the existing database, it was not possible to save sequences in a meaningful way. Technically, it may have been possible, but this was by means of a BLOB (Binary Large Object) in the database, invalidating all the advantages related to using a relational database.

The following requirements were derived from the general design process and by talking with the other groups:

- It should be possible to nest sequences, but it is not required that more sequences are nested on the same level. (requirement for Sequence Viewer and Week Schedule).
- The position of each pictogram should be stored in the database (requirement for Life Stories)
- It should be possible to save a collection of pictograms to a single frame, such that the user must choose a pictogram when pressing that frame. (requirement for Life Stories).
- Each sequence must have a name, a pictogram and a profile attached to it (requirement for Life Stories)

These requirements resulted in the database design shown in Figure 4.1. Attributes that are not self-explanatory in the figure, are described below.

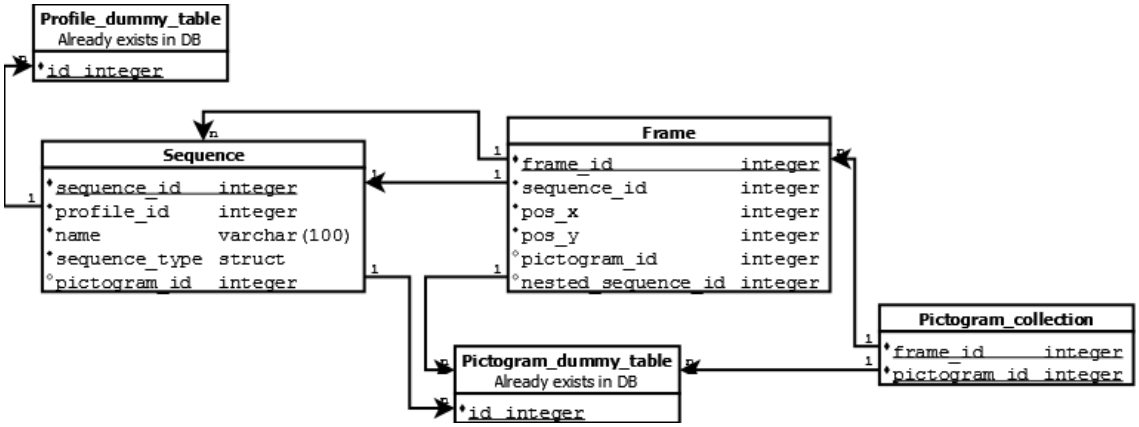


Figure 4.1: Relational database diagram of the tables used for saving sequences. Note, that the arrows are pointing at entire tables and *not* specific attributes.

Sequence:sequence_type
: This is an an identifier denoting which app this sequence belongs to. It is of type `enum` in the diagram to communicate the intention of this attribute to the database group. The database implementation was done using the `int`

data type and then providing the current app name in the API (to associate the sequence with the correct app).

Frame:pos_x / Frame:pos_y
The relative position of the pictogram. They are not intended to be actual coordinates, but merely a relative position on a frame.

Frame:nested_sequence_id
Contains the ID of the nested sequence. If this frame does not contain a nested sequence, this attribute is null.

Pictogram_collection
Every Frame can have a pictogram collection associated with it, and by the design of this table, every pictogram collection can have every pictogram assigned only once per Frame.

The other subject of collaboration was the GUI design described below.

4.2.3 GUI Design Collaboration

It made sense to unify the GUI design of the Week Schedule and iOS Visual Scheduler projects, to give the customers a common feel of the apps. The GUI design of the Week Schedule was derived through prototypes and collaboration with sw610f14. Most of this collaboration was conducted through meetings. Collaboration was done solely with the mentioned group, as development of the Web Schedule had not yet started.

Current Week Schedule in Institutions

The point of creating the Week Schedule app was to replace currently used bulletin boards and the related difficulties of managing a large number of pictograms. Figure 4.2 shows a picture of the current week schedule and it is easy to imagine the amount of work involved in clearing the entire board of pictograms.

Figure 4.2: The currently used week schedule.

Figure 4.3 shows what is currently used when a citizen can choose between different activities.

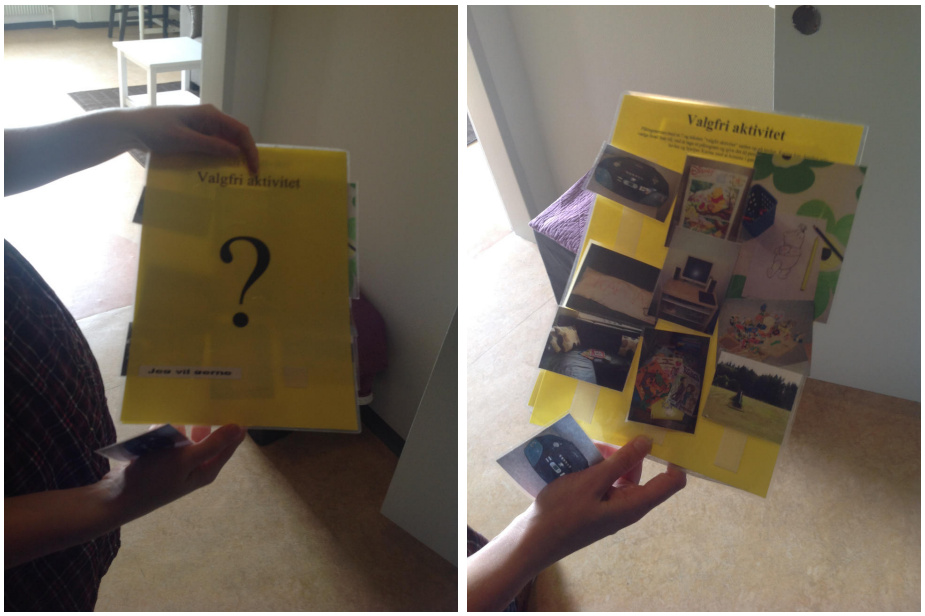


Figure 4.3: The current way of denoting that the citizen can choose between different activities (left) and how a citizen is shown the activities he/she can choose from (right).

Some prototypes were made, based on the pictures above. These prototypes can be seen in Appendix B.

Customer Meeting

Through the project's customer contact person, a meeting was arranged with some of the customers. The customers were Mette Als Andreasen and Kristine Niss Henriksen, both employed at the institution *Birken*. They were presented with the prototypes of the Week Schedule's two modes, guardian and citizen mode (see Section 5.4). Based on the presentation of the prototypes, they provided feedback regarding improvements to the proposed design and functionality, which was put together to form the following requirements:

- In the case that a frame has multiple pictograms associated with it, (i.e. when the citizen has the option to choose between a number of different activities), a pop-up should appear when the frame is touched, marked with the same symbol as the one that appears on the frame.
- The entire background of each week day in the planner should be colored according to the international standard.
- The current day should be highlighted.
- Finished and coming activities should be shown, along with a check mark to the left of each pictogram, indicating whether the activity is finished or not. The visibility of finished activities should be changeable with a setting.
- It should be possible to move between weeks by swiping or clicking arrows.

- The guardian should be able to adjust how many days the citizen is shown at a time (1 day, 3 days, full week etc.).
- The week planner should never display only part of a pictogram to a citizen.
- The app should be usable in both landscape and portrait mode.

As the Week Schedule was implemented from scratch in this project, a light weight analysis of the usage was in order. This analysis is described in the following section.

Use Cases

The following describes some use cases for the Week Schedule app. The use case diagram sown on Figure 4.4 is based on the prototypes and the requirements given by the customers. In the use cases activities and pictograms are the same.

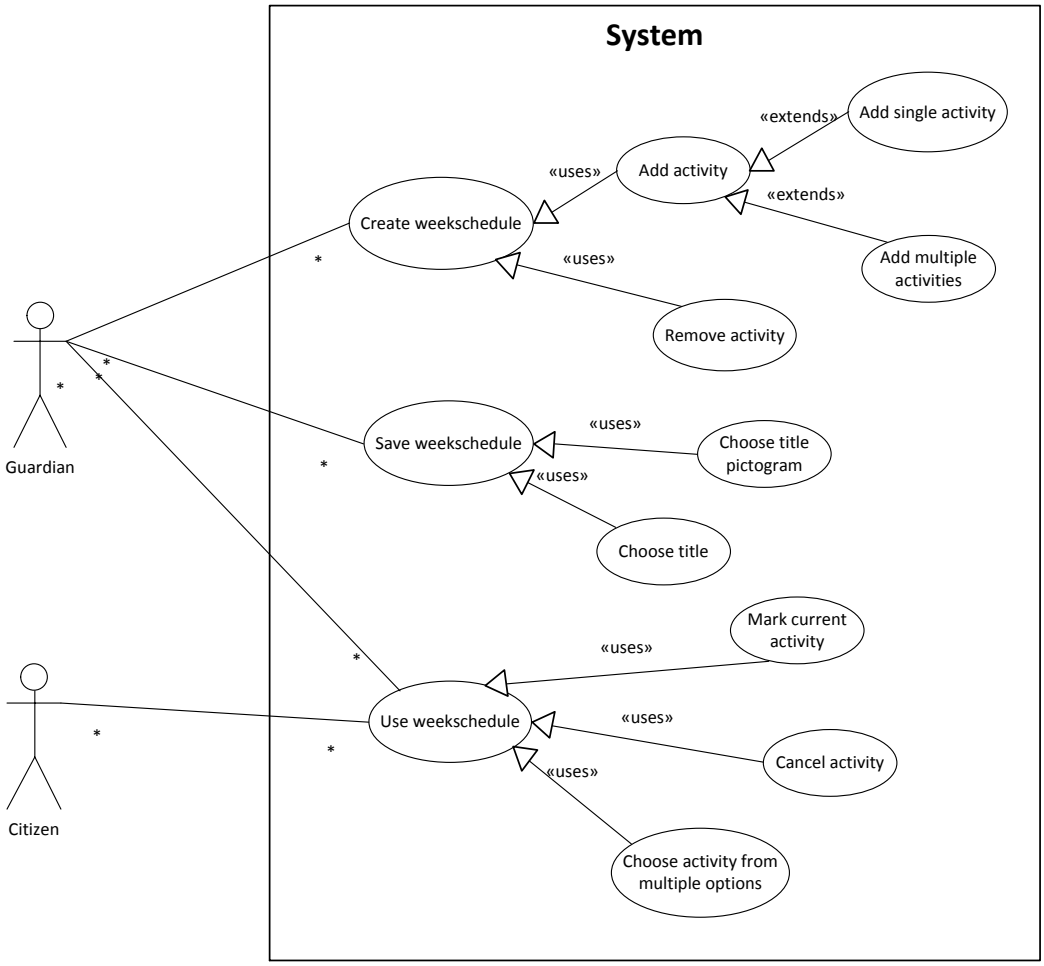


Figure 4.4: Use case diagram of the Week Schedule app.

Create week schedule
The guardian presses the button for adding a new schedule. The edit mode opens and every day in the week contains a plus sign. When a plus sign is pressed, the PictoSearch app opens and the guardian can choose one or more activities. If more than one activity is chosen, the picture appearing in the schedule will be denoting an option. If only one activity is chosen, it is displayed directly in the schedule. To delete an activity, the guardian must long press that activity.

Save week schedule
The guardian must choose a name for the schedule by tapping the text at the top of the schedule. A title pictogram must also be chosen by pressing a button in the upper left corner. When both are chosen, the schedule can be saved by pressing the button with an image of a disk.

Use week schedule
An activity can be marked as current simply by pressing it. An activity can be cancelled by long pressing it. To choose an activity from a frame with multiple activities, the user must press that activity and a dialog box appears with the available activities for that frame. When an activity is chosen for a frame with multiple choices, the selection can not be changed.

4.2.4 Pictosearch and Remote Database Issues

Near the end of sprint 4, a collaboration occurred between our group, sw605f14 (Launcher), sw612f14(Remote database) and sw615f14 (Pictogram to Speech) in order to resolve some issues with the pictogram searching app Pictosearch. Pictosearch broke down completely when the local and remote database began to synchronize due to the increased amount of pictograms retrieved from the database by PictoSearch. There where two main issues to be looked at: the way PictoSearch handles memory and the way it searches in the database.

The former was a reappearance of an issue reported in sprint 2, where PictoSearch would crash due to an out of memory exception, when reopened repeatedly. The difference between after it was fixed and before sprint 2, was that the crash would happen the second time Pictosearch was used to search for a pictogram within an app, when limiting the database to containing just 100 pictograms. This issue was resolved by improving handling of the activity life cycle of PictoSearch and by increasing the apps heap size. After the improvements, PictoSearch was stable even with the full 22,000 pictograms in the database, but it would still take minutes to perform a search with Pictosearch, this brings us to the second main issue:

Pictosearch performs its searches by loading all pictograms, including their associated data such as sound and bitmap, from the local database into memory, and it then searches on names and tags to rearrange the pictograms. This was too slow when dealing with the full amount of pictograms had to be changed in order to make the pictograms available for other apps in a reasonable amount of time. This was solved by adding a new method to the local database library Oasis-lib to perform

substring searches. This allowed PictoSearch to limit the amount of pictograms loaded into memory and run much faster. As an additional limit, 100 pictograms was added to ensure PictoSearch would run smoothly even on a search string with a potentially large result like searching the letter “e”.

5 Design and Implementation

This chapter contains descriptions of central workings in the source code. The first section is a description of a single class, that was implemented from scratch during this project-period. It is included because it provides evidence of an important choice that was made during the project.

5.1 Roadmap

The primary focus for each of the four sprints was as follows:

Implementation of the DBController
: The DBController manages translation between objects of the local Tortoise sequence class and objects of the database sequence class.

Life Stories
: This builds on the app from 2013, but has been changed to fulfill more of the customers' requirements.

Week Schedule
: A new app with some features based on the ones already existing in the project.

5.2 Implementation of the DBController

The responsibility of the DBController class is to convert back and forth between Life Story Sequence objects and Sequence objects returned by the database API, OasisLib. The optimal solution would have been to use the OasisLib Sequence class directly, but an attempt to do this revealed a vast amount of refactoring throughout the code. The amount of refactoring was deemed too comprehensive and thus the solution with the DBController as a translator was chosen.

The DBController is implemented using the singleton pattern. This ensures, that there will never be more than one instance of the class at any time, and further that this object can be easily referenced where needed.

A method, `loadCurrentCitizenSequences()`, is called when the guardian or citizen needs to load all sequences belonging to a specific citizen. It takes three parameters: The citizen's profile id, the type of the sequence (schedule or life story), and the current application context. This method invokes a cascade of methods, all designed to translate Sequence-objects from OasisLib to Life Story Sequence-objects. It updates the list of stories available on the current citizen profile in the app. The order of method calls is displayed in Figure 5.1.

Inv. order	Method	Parameters	Return value
1	loadCurrentCitizenSequences(...)	profileID, sequenceType, context	void
2	morphDBSequenceListToSequenceList(...)	dbSequences, context	sequences
3	morphDBSequenceToSequence(...)	dbSequence, context	sequence
4	morphDBFramesToMediaFrames(...)	dbFrames, context	frames
5	morphDBFrameToMediaFrame(...)	dbFrame, context	frame

Figure 5.1: Invocation order when loading all sequences associated with a specific citizen.

When saving sequences to the database, translation from Life Story Sequences to database Sequences is conceptually performed in the same way. The next section describes some specific implementation details on the Life Stories app.

5.3 Life Stories

The changes in the Life Stories app was based on the following three factors: The sections on usability testing and future work in the previous Tortoise report [5], the new requirements specification as shown in Appendix A, and feedback from the customers during meetings. Some of the features written as the goals for sprints 1 and 2 were later discarded, and are not included here.

The following is a list of the changes that were introduced:

- **GUI changes:**
  - Sequences of pictograms have the same size and are locked in a line.
  - The menu bar to the right is moved to the top in order to achieve consistency in the app.
  - The options for adding content to a frame is displayed in a dialog box.
- **Feature changes:**

  - The original view (citizen) mode was removed and a new one was implemented by sw607f14 (Sequence).

- **Added features:**
  - A customizable icon for choice-frames (when more pictograms are associated with a single frame).
  - Send email containing life story.
  - Editing existing life stories.

5.3.1 GUI changes

The GUI was changed considerably. Figure 5.2 shows the difference between the old and new edit (guardian) mode. Previously the frames could be added without content and placed anywhere on the screen, but now frames are added in a line and automatically removed if the content is deleted. As seen, the menu has been moved to the top, freeing more space on the screen for the life story. If the life story becomes longer than the view, it is possible to scroll through it.

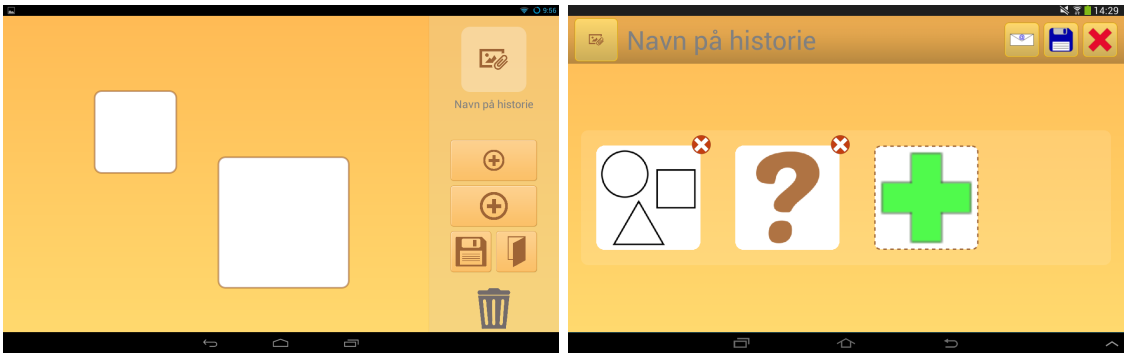


Figure 5.2: Old (left) and new (right) edit mode.

Figure 5.3 shows how the frame content is managed in the old and new version. If more pictograms were associated with a frame in the old version, the main frame would show a white background with a small gray text, e.g. Valg 1, and the associated pictograms would be shown on the right menu bar, in the area above the button with a plus-sign. In the new version the choice-frame is replaced with a question mark by default, but the user can choose a different icon. The associated pictograms are shown with the new choice-frame icon in a dialog box.

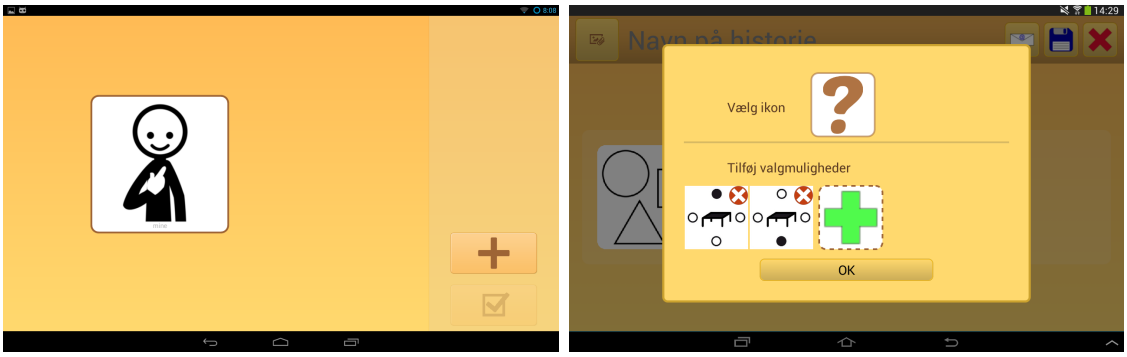


Figure 5.3: Old (left) and new (right) frame content.

Previously when saving, the user would be asked to choose if the life story should be saved as a template, for the guardian to reuse, or a story. This functionality has been changed, such that it is no longer possible to save a life story as a template. Instead it was made possible to edit an existing life story. This was previously only possible on templates. Figure 5.4 shows the new simplified dialog box compared to the old one, which is shown when the user clicks the save button.

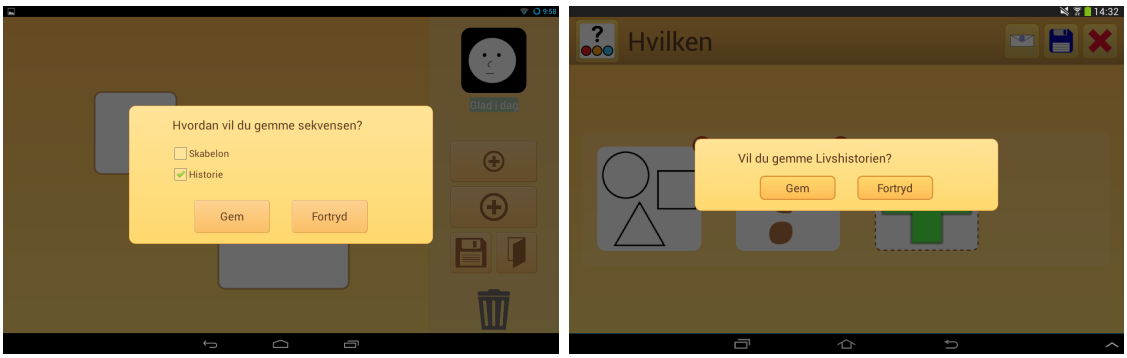


Figure 5.4: Old (left) and new (right) edit mode save dialogue box.

Figure 5.5 shows the changes to the main view, where saved life stories are displayed. Minor changes have been made to the colors and buttons, so they are consistent with the other apps. These changes were made simply by using the common GComponents instead of our own colors and buttons. Previously, clicking the profile icon (top left) did nothing, but now it allows the user to change citizen profile. Clicking the plus still adds a new life story. Clicking the wrench will activate edit mode, which displays the delete button (as shown on the old screen), but now also allows the user to edit the life story. The guardian/citizen switcher (two opposing arrows, shown on the old screen) has been removed and the functionality has been moved to the profile selector in the Launcher app. By customer request, an exit button has also been added.

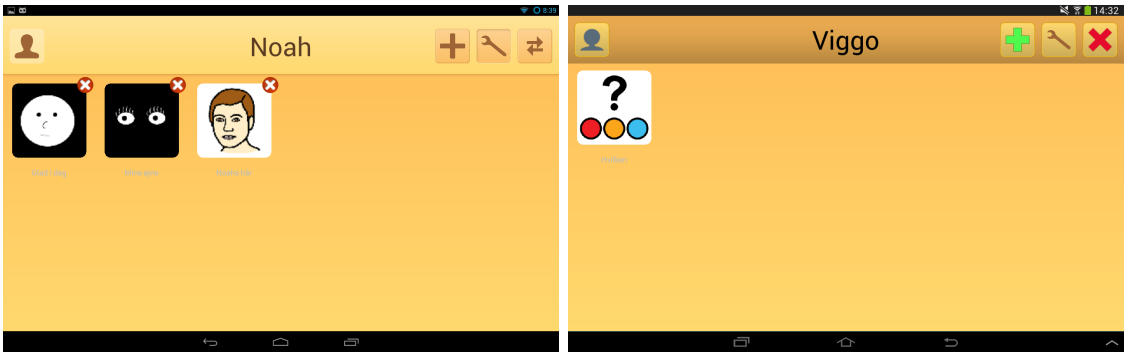


Figure 5.5: Old (left) and new (right) main view.

The view (citizen) mode has been removed from this app, and replaced by a new sequence viewer made by sw607f14. The sequence viewer is launched when clicking on a life story in the main view when the app is not in edit mode.

Implementation of the GUI

According to requirements frames in a life story must have the same size and be locked in a grid. This was accomplished by replacing the initial drag and drop functionality with an AdapterView, originally implemented by last year's Zebra (Sequence) group [4]. Using the AdapterView changed the procedure for rendering frames. Previously, all frames was re-rendered whenever there was a change in the list. Using the notify method on the AdapterView, caused only changes in the list to be re-rendered.

Moving the menu bar was simply a matter of changing the xml-file responsible for the layout, but the dialog box for editing the content of a frame was made from scratch. When a frame is clicked, it calls `renderAddContentMenu(int position)` with the postion given by the `onItemClick(...)` method from the AdapterView, so the content corresponding to the frame on the given position will be shown in the dialog. This position is also used when changing the content, to make sure the right frame is altered.

5.3.2 Feature - Choice Icon

The choice icon is the picture shown on a frame, when more than one pictogram is associated to it. If no specific pictogram is chosen, the `pictogram_id` associated with a frame (see Figure 4.1) is `null` and a default picture is set (Figure 5.3 shows the default icon). If this icon is clicked, PictoSearch is launched, and the user is asked to choose a pictogram. This is then saved on the frame using `setChoicePictogram(Pictogram pictogram)` and a delete button is shown it the dialog. Clicking this will call `removeChoiceIcon(...)`, setting the `pictogram_id` to `null` again.

5.3.3 Feature - Editing Existing Life Stories

To do this, the intent opening the edit mode contains an integer, corresponding to the position, that the life story has on the list of life stories on the current citizen. If this number is not provided, a new life story is created. This is very similar to the way templates were used before.

5.3.4 Feature - Send Email with Life Story

When using this functionality, the user is initially asked if the life story should be printed vertically or horizontally. The dialog box for this is displayed in Figure 5.6. When *Godkend* is pressed, the `printSequence()` method, which is described in Figure 5.7, is invoked and this invokes the methods described in Figure 5.8.

Figure 5.6: Screenshot of the print alignment dialog.

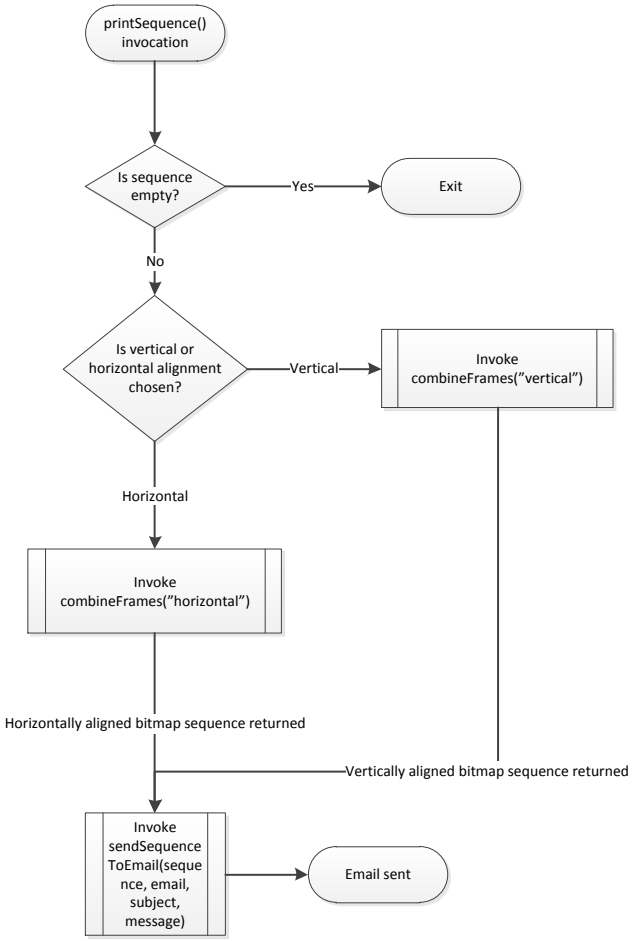


Figure 5.7: Flowchart of the printSequence method.

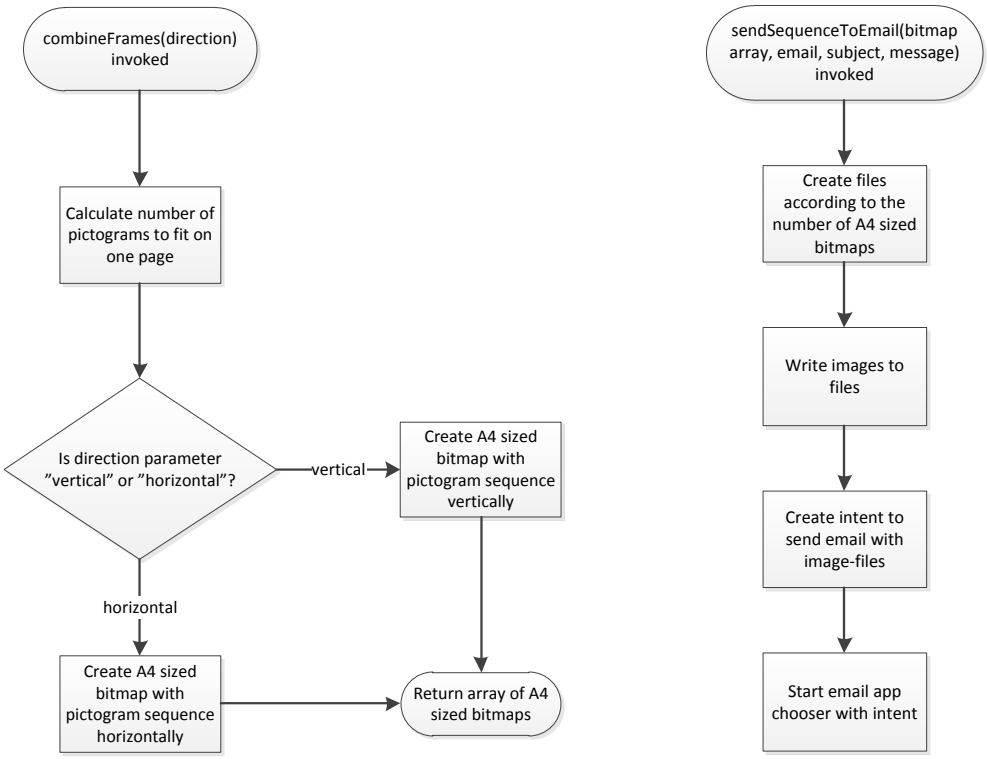


Figure 5.8: Flowchart of the combineFrames and sendSequenceToEmail methods.

The next section contains a description of the implementation details for the Week Schedule app.

5.4 Implementation of Week Schedule

The Week Schedule app is a tool for guardians to organize a citizens week. This section details some of the nontrivial parts of the implementation, while describes the analysis.

5.4.1 Guardian Mode

When the guardian mode part of the week planner is started, the first thing that happens is that the buttons for adding new pictograms to a specific day, are added to the required views. This is done using a method called `renderSchedule()`. It is dynamic since it works by traversing the parent layout of each week. After locating the correct layout, `addButton()` is called. The method returns a clickable ImageView with the add icon and sets a click event handler which starts PictoSearch when the view is clicked. The `showAddButtons()` method then adds these new image views to the week schedule.

When the user returns from PictoSearch and the Week Schedules `onActivityResult()` method is called, the add button is first removed, the pictogram(s) received through the intent from PictoSearch is added to the correct day, using a variable indicating the selected day which is set when an add button is pressed, and lastly the add button is added again, to make sure it is always on the bottom. Figure 5.9 shows how the guardian mode looks after adding some activities.

Figure 5.9: Week Schedule in guardian mode.

Every time a frame is added to a view, this frame is also added to an array of sequences, which is used to reconstruct the schedule when something is changed, and to save the schedule in the database. To delete a given frame it can be long pressed or each associated pictogram can be deleted in the content editing dialog. This dialog is implemented in the way described in Section 5.3. The top menu is exactly the same as in the Life Stories app, except there is no option for sending a schedule as an email, as this was not initially requested by the customers.

5.4.2 Citizen Mode

The citizen mode only has a few differences compared to guardian mode. The add buttons and save button are not shown, and the scroll views have been locked, to prevent showing half pictograms. This was done on request from the customers, as it may distract some citizens. Instead arrows are shown, when there are more pictograms to show. Clicking an arrow will move the view exactly one pictogram up or down.

Clicking a frame on the day marked with a black border (the current day), will mark the frame with a similar border to indicate this is the current activity. Clicking a frame with a question mark (or custom choice icon) will open a dialog similar to the content editing dialog, but without the delete buttons and add button. Clicking one of the pictograms shown in the dialog will close the dialog and replace the question mark with the clicked pictogram.

Figure 5.10 shows how the scedule from Figure 5.9 looks in citizen mode.

Figure 5.10: Week Schedule in citizen mode.

6 Test

The testing during this project was confined to debugging when developing and integration- and usability-testing. The following section describes the integration-testing.

6.1 Continuous Integration Testing

It is particularly important that every project in the multi-project builds correctly, because of dependencies between the subprojects. If a dependency for a project fails to build, then it is likely that the project depending on it also fails.

To ensure integrity of the APK's available of each subproject, after introducing new changes to the code, a continous integration testing system called Jenkins was used. Every time a new version of one of the projects' APK's was available on the projects' main git branch, Jenkins would try to build it. This ensured that the available APK's built without issues and if not, an administrator was notified by the system.

6.2 User Test

User tests were conducted with two of the guardian users using the IDA (Instant Data Analysis) method [3]. The users were asked to perform a set of predefined tasks (see Appendix C.1), designed so they introduce the functionality of the apps and then test the users ability to access this functionality. One such task is *Cancel a scheduled activity*, which hint the existence of the cancel functionality, but the user have to apply intuition to figure out how to do it. Each user were interviewed before and after the test, regarding general experience with computers and tablets, and the impression they got from testing the apps (see Appendix C.3).

6.2.1 Usability Issues

Usability issues are categorized into cosmetic, serious and critical severity. The following usability issues were identified.

Critical

- In the Week Schedule viewer the way an activity is flagged *Cancelled* was not intuitive for the test persons, both mistakenly closed the viewer and only one of the them was able to perform the task.
- In the Week Schedule editor, a crash happened when retrieving pictograms from PictoSearch.
- In Life Story editor, a crash occurred when attempting to email the life story.

Serious

- All: The red cross on the exit button was confused with deleting frames and cancelling frames multiple times.
- Both editors: Test persons repeatedly added pictogram frames with a choice when asked to add multiple frames with a single pictogram.
- Both editors: It was not obvious how to change the pictogram viewed on choice frames.

Cosmetic

- Both editors: The icon for selecting title pictogram was not initially recognised by the test persons.

6.3 Result of Testing

Continuous integration testing was useful to our group mostly in form tracking dependencies, though it was not always up to date due to varies development branched on git. Only one critical (email crash) and one serious (exit button) usability issue was corrected before the final sprint end meeting. The other issues are subjects of future work.

7 Future Work

This chapter is divided into three sections: Bugs, Refactoring and Features. The order of which they are mentioned in also expresses how we feel they should be prioritized. Before adding new features – even obviously missing ones, the code should be refactored, as adding new features is more difficult and may introduce more bugs if they are implemented before a thorough refactoring process. A second argument is that, when new students receive the source code, refactoring forces them to investigate the implementation thoroughly before introducing new features, which improves the chance that the features are implemented in an appropriate way and avoid unnecessary complexity.

Before looking into bugs, the justification of the Life Stories app should be considered. It resembles the Sequence app to a degree, and it should be decided whether to discontinue one of the two apps, or if they should be merged.

7.1 Bugs

These bugs should be fixed before refactoring:

- Life Stories
  - The Life Stories app does not make use of a SavedInstanceState object. This has the unfortunate side effect, that a Life Story may disappear if either another app is opened and the Android system needs the memory, or if the app is running on a device that has a lock screen in portrait mode. This forces the app into portrait mode for a short period of time after the screen is unlocked.
- Week Schedule
  - If you remove an activity from any day, it seems like it is removed correctly, but when the schedule is saved and loaded again, it shows that it

is always an activity from Monday that has been deleted in the database. This bug appears when editing an existing schedule.

7.2 Refactoring

During this project period, some minor refactoring was done when so-called bad smells were encountered, but major refactoring was avoided in general due to time constraints and the fact, that the main focus was on delivering functioning apps.

Before anything else, the todos in the source code should be fixed. They should all at least be evaluated – most of them are code that should be refactored. Afterwards, decide whether `onClick` listeners should be implemented in the code or defined in the XML-files using the `onClick` attribute and make it as consistent as possible throughout the code.

Another issue is the fact, that the apps are not using the Sequence class provided by OasisLib. It was deemed less time consuming to simply translate between the OasisLib Sequence class to the Life Stories/Week Schedule Sequence class. This should be refactored to use the OasisLib Sequence class, so the translation is avoided when saving and loading sequences.

A part of the project's package structure should also be re-evaluated, as the current is somewhat inconsistent.

In general, the bad smells[2] to look for is mostly of the following types:

- Long Methods
- Large Classes
- Duplicated Code
- Shotgun Surgery
- Switch Statements

After these areas of refactoring have been considered, new features can be implemented.

7.3 Features

Some requested features were not implemented during this semester. The missing features are:

- Week Schedule:
  - It should be possible to hide some days on the schedule.
  - The action required to cancel an activity should be changed from long pressing to dragging a red cross from the menu bar to the activity.
  - The symbol marking the current activity should be changed to an arrow pointing at the activity or a green check mark next to the activity.

  - It should be possible to change the order of activities on a day by drag-and-drop, the same way the order of pictograms can be changed in the Life Stories app.
  - Arrows for scrolling over activities should disappear when irrelevant.

- Life Stories:
  - Pictogram text should be visible in citizen mode.

- General:
  - Ability to copy a week schedule/life story from one citizen to another.
  - Only icons from GIRAF_Components should be used.
  - All dialog boxes should be replaced with dialogs from the GUI project.

8 Conclusion

Institutions taking care of citizens with ASD utilize a number of tools when communicating with and educating the citizens. Most of these tools involve use of pictograms which, in the case of many citizens, are specialized to accommodate their inability to classify objects independent of insignificant attributes as exemplified by the red/yellow bus issue mentioned in Chapter 1. Managing these physical pictograms is time wasted, that could be better spent on the citizens.

To minimize this waste of time, the pictograms could be organized in a database and the different tools they use in conjunction with the citizens, organized as a set of apps for the guardians and citizens to use on tablets. This particular project was focused on creating a Week Schedule app and continue development on an existing app, Life Stories, for citizens to tell stories about what they experience.

The overall project was managed using Scrum, but because the group was relatively small, little formal ceremony was needed in the group. Knowledge sharing was, however, taken very seriously. Knowledge sharing and overall project management in the multiproject was mainly done using the Redmine project tool and weekly status-meetings.

In the first sprint, nothing significant was developed, but time was spent on getting to know and comment the existing code. This continued in the second sprint where the existing code was modified to start using the new database design released by sw614f14 as well as the common GUI elements released by sw603f14. Some problems occured with another app, PictoSearch, which was temporarily the main focus. Aside from this, development started on the Week Schedule in collaboration with sw610f14 and on an extension of the database to accomodate sequences better in collaboration with sw607f14 and sw614f14.

In the third sprint the collaboration with sw607f14 continued with them developing a common sequence-viewer and us finishing the new database design. An early version of the Week Schedule app was finished, and the interface of the Life Stories app was simplified. Additionally the adaption to the database was finished.

In the fourth sprint, citizen-mode was implemented in the Week Schedule app

along with marking of the current weekday, the possibility of marking an activity as ongoing, and the feature of scrolling activities when there is too many to fit on the display. It was also made possible to cancel an activity in citizen-mode. Both loading and editing schedules was also implemented in this sprint. On the Life Stories app, the features of loading, viewing and editing of sequences was completed and a feature for sending an email containing a life story was implemented.

By the time of the final sprint end meeting, both the Week Schedule and Life Stories apps were completed with the features mentioned in this report and only a few known bugs.

Testing was confined to usability tests, automatic integration tests using Jenkins and debugging when implementing. Taking into consideration the limited testing procedures, some bugs will inevitably surface when the apps are used on a daily basis. Consequently, it is important for the students who continue the work on these apps, to start the project by collecting bug reports from the customers who have been using the apps.

At the initial project meeting, the students were advised by the semester coordinator to focus on finishing the apps completely in order to be able to deploy them on the customer tablets at the end of the final sprint. This means that very little refactoring has been done and thus, refactoring should be a main focus for future students working on these apps.

9 Reflection on the Multi-project

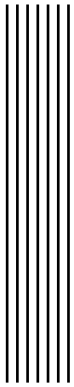
Working in a multi-project setting was new for most of the students. Communicating between the groups had to be learned along the way. A positive gain of working in a multi-project is that a lot can be accomplished when all the groups are working together, since tasks can be delegated to different groups instead of just having one group do it all. E.g when we had a change request for the database we did not have to spend time implementing it ourselves, but could leave to the database group and we could focus on other issues in the meantime. This had the disadvantage that it could take a long time for the groups to respond to requests, since they could have other requests or things to do.

We discovered that being many people also proses some problems. One problem we encountered this semester, that many of the apps relied on the functionality of a single app. This in itself is not a problem, but the group who were responsible for the app were hard to get in contact with and during the project a lot of their features were breaking and they did not always fix them. Because of this it would be advice to make sure that people responsible for vital apps are easy to reach, responsive and responsible. A solution could be a person who keeps track of all the groups progress in relation to their goals, and the needs of the other groups in the multi project, though experiences from last year students suggest that this is difficult.

It is recommenced start communication with groups that are relevant to what you are doing in the very beginning of the project. E.g if you have something which could be nice to have in the database, immediately go and consult the database group and let them consider your request.

All in all, we find that the multi-project was a great exercise in communication with others and generally working in a bigger setting than just a small group. It was a learning experience and has equipped us to be better at potentially working

on projects in a big setting.



Bibliography

[1] American Accreditation HealthCare Commision (2012, May 16). Autism. http://www.ncbi.nlm.nih.gov/pubmedhealth/PMH0002494/ - Accessed May 12, 2014.

[2] Beck, K. and M. Fowler. Refactoring - code smells. http://sourcemaking.com/refactoring/bad-smells-in-code - Accessed May 20, 2014.

[3] Kjeldskov, J., M. B. Skov, J. Stage, and F. B. Vej (2004). Instant data analysis: Conducting usability evaluations in a day. In *in Proceedings of the Third Nordic Conference on Human-Computer Interaction*, pp. 233–240. ACM Press.

[4] Knudsen, S., C. Lykke, S. Enevoldsen, and S. Olesen (2013, May 30). *GIRAF: Zebra.*

[5] Mortensen, C., B. Holbech, and N. Čokljat (2013, May 30). *Tortoise - An application for managing autistic children's life stories in GIRAF.*

[6] Nilsson, A. S., M. T. Pedersen, N. B. Pedersen, and K. Plejdrup (2014, May 28). *Kategorispillet.*

A Specification Requirements

Requirements relevant for Week Schedule and Lifestories. The complete list can be found in the report by [6].

A.1 Requirement Prioritization

The requirements in this section is split into the different application they belong to. Each individual requirement has a number in end, which shows the priority of this specific requirement.
(1) - Requirements that must be fulfilled.
(2) - Requirements that must be fulfilled, but which are not essential.
(3) - Requirements that must be fulfilled, but only if there is additional time available.

A.2 Profiles

1. Guardian profile:
    a) Has access to all functions in all applications. (1)
2. Citizen profile:
    a) Has access to functions and applications, which are authorized by a guardian profile. (1)
    b) As standard, citizen profiles does not have access to any functions. (1)
    c) Citizen profiles can never use the three androids buttons (back, home and menu). (1)
3. Shifting between profiles requires approval, e.g. by the use of QR code. (3)

A.3 General

1. No applications must close, stop or crash unexpectedly. (1)
2. All applications must synchronize with the global database, once a day, if there is access to the Internet. (1)
3. All applications must run in landscape mode (horizontal). (1)
    a) Sequences and Week Schedule must be able to use portrait mode. (1)
4. All applications, except the Launcher, must have a 'close' button. (2)
    a) When the button is pressed, it must close the application, if the user has permission to close applications. (1)
    b) If the button is pressed and the user does not have access to close the application, the application must be able to close with the help of a guardian, e.g. by scanning QR code. (1)
5. When and only when there is a permitted input, there must be some kind of response, which makes sense in context, but otherwise this is up to the individual developer. (2)
    a) Applications with functions targeted towards citizens must only use singleclick, drag and drop, and swipe input. (2)
    b) If there is a function, which should only be usable by a guardian, then all Android gestures are permitted. (2)
    c) All applications must be named with meaningful Danish names. (1)
6. All applications must be able to play a pictogram's associated sound. (2)

A.4 General GUI

1. The general color scheme must be black and white. (3)
    a) It must be possible to choose a color theme for each application. (2)
    b) Each profiles must be able to have their own chosen color theme. (2)
2. All applications must use the same GUI components. (2)
3. All applications must indicate if there is a sound attached to a pictogram. (2)

A.5 Sequences

1. Citizen profiles must be able mark how far they are in a sequence. (1)
2. The guardian can change whether sequences are shown horizontally or vertically for the citizens. (2)
3. Guardian profiles must be able to view, create, edit and remove sequences. (1)

4. There must be multiple ways to view sequences. (1)
    a) It must be able to show one pictogram, three pictograms or a complete sequence. (1)
5. It must be possible for a guardian profile to authorize a citizen profile to be able to make sequences. (2)

A.6 Life Stories/Week Schedule

1. Citizen profiles must be able to mark how far in a sequence they are. (1)
2. Colors for the week schedule must follow the international color standard for days. (1)
    a) These colors must always be viewable, even if you have scrolled down in a sequence. (3)
3. It must be possible to turn on a feature, such that days are switched upon swiping to either side, when you are in the day tasks. (1)
4. This application must function as a tool to plan daily activities for citizens. (1)
5. This application must be able to help citizens to formulate their day in pictograms. (2)
6. This application must be able to allocate periods where citizens can choose between preselected activities. (1)
7. Guardian profiles must be able to define how many pictograms are shown at a time. (2)
    a) It must be able show one pictogram, three pictograms or a complete day. (2)
8. Sequences of pictograms must have the same size and be locked in a grid. (1)
9. When a citizen has a choice to make, other functions must be locked, until a choice has been made. (2)

B Protoypes for the Week Schedule

The following are the prototypes presented to the customers on the meeting about the requirements for the Week Schedule app.

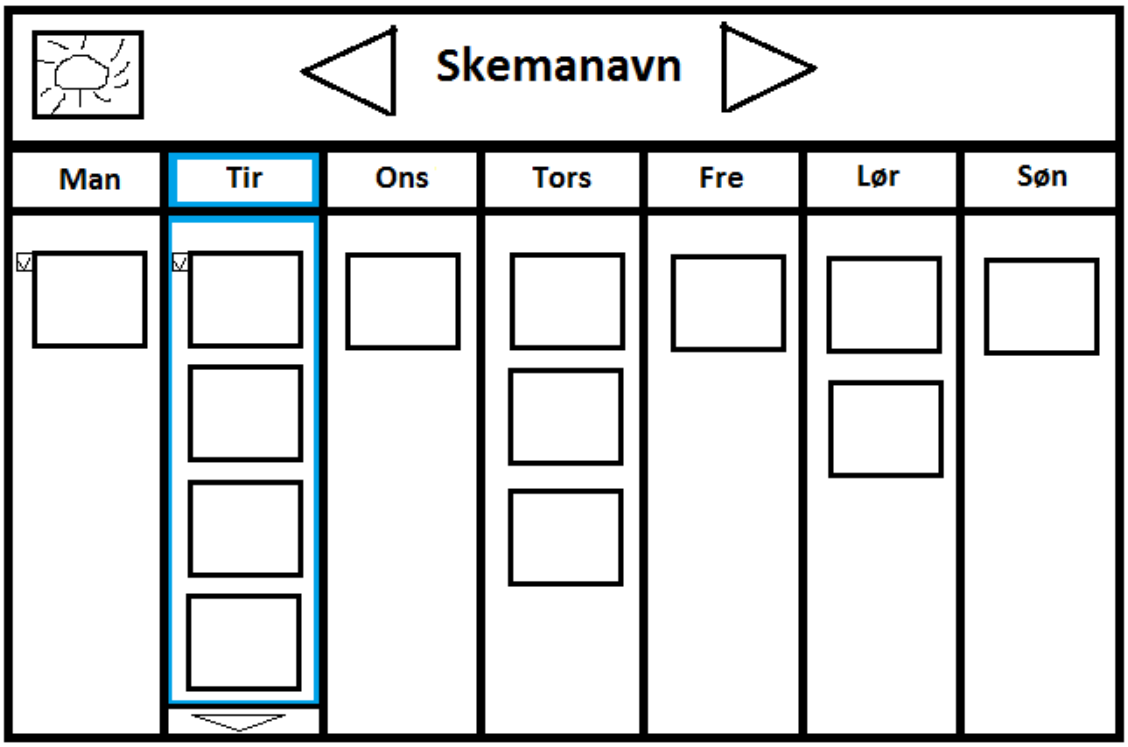


Figure B.1: Prototype of the week planner in citizen-mode.

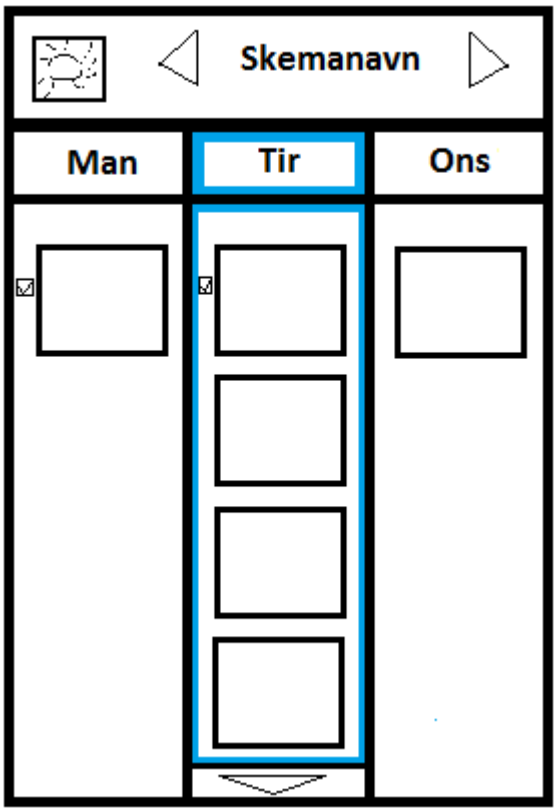


Figure B.2: Prototype of the week planner in citizen-mode in portrait orientation.

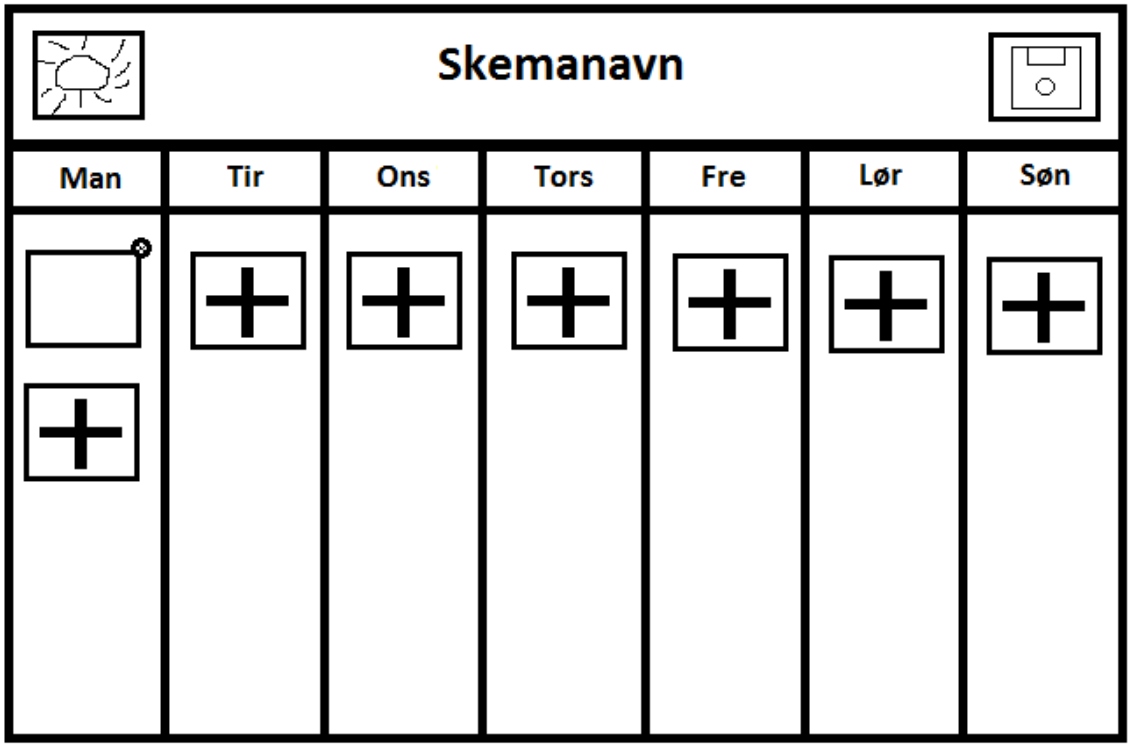


Figure B.3: Prototype of the week planner in guardian-mode.

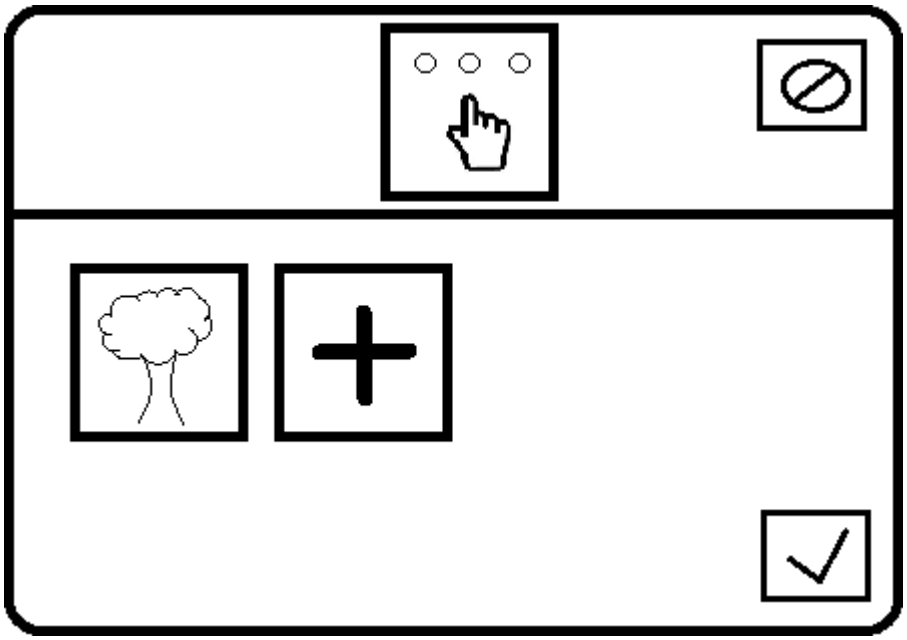


Figure B.4: Dialog box used both in the week-planner and Life Stories.

C User Test

C.1 Week Schedule

- Create a new week schedule with at least two activities on the current day and one on another day. Make at least one of activities a choice with multiple pictograms.
- Choose the activity with most pictograms and replace one of them with another.
- Change the pictogram displaying the choice in the schedule.
- Delete an activity from a day.
- Save the schedule .
- Close edit mode.
- Open the schedule just saved and mark an activity and ongoing.
- Cancel an activity.

C.2 Life Stories

- Create a new life story.
- Add three pictogram to the life story.
- Change the order of the pictograms.
- Delete a pictogram.
- Save the story.

- Send the story via email.
- Close the story and open the viewer

C.3 Questionnaire

C.3.1 Table Experience

- What are your general experience with computers.
- What are your general experience with tablets.
- What are your general experience with Android tablets.
- What are your general experience with the GIRAF project.

C.3.2 Impressions

- How was the overall impression of the apps?
- Do you see a use for the apps in daily activities?
- Was there any part of the apps you found confusing?
- Was there anything that did not live up to your expectations?
- Anything you would like to change regarding the app?

D Resumé

Dette projekt var en del af et større multi-projekt, hvor 16 grupper af studerende arbejdede sammen med det fælles formål at udvikle apps til brug for mennesker med autistisk udviklingsforstyrrelse. Projektet blev styret vha. Scrum og det samme var gældende for det overordnede multi-projekt.

Denne rapport dokumenterer udviklingen af to apps, som skal gøre livet nemmere for mennesker med autistisk udviklingsforstyrrelse og deres omgivelser. Svære grader af autisme kan medføre en mangel på talesprog som gør, at al kommunikation må foregå via små billeder - bla. a. piktogrammer, der illustrerer ting og handlinger. De sættes sammen til sekvenser for at kommunikere behov og ønsker og bruges også til at lave dags- og ugeskemaer. Mængden af disse billeder gør dem besværlige og tidskrævende at håndtere.

Det specifikke formål med de to apps, Ugeplan og Livshistorier, er at de skal fjerne behovet for fysiske billeder/piktogrammer. Begge apps benytter en database med mange tusinde piktogrammer, som let kan findes via en tredje app og bruges til at sammensætte skemaer og sekvenser.