

Android Timer Application





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

Title: Timer Application
Subject: Android Systems
Semester: Spring Semester 2012
Project group: sw602f12

Participants:
Kristian Kolding Foged-Ladefoged

Rasmus Hoppe Nesgaard Aaen

Simon Blaabjerg Frandsen

Supervisor:
Ulrik Mathias Nymann

Number of copies: X

Number of pages: X

Number of appendices: X Pages

Completed: X

Synopsis:

This project is about the development of...

Preface

This project has been produced in the spring of 2012 in the sixth semester of the software engineering study at Aalborg University.

Contents

I	Introduction	2
1	Analysis	4
1.1	System Definition	5
II	Development	7
2	Project Management - on a Multi-Project Level	9
2.1	Modification of SCRUM	9
2.2	Versioning	10
3	Design	11
3.1	Vision	11
3.2	Use Cases	13
3.3	Prototyping	13
3.3.1	Metaphors	14
4	Development Process	16
4.1	Sprint Walk-through	16
5	Implementation	19
5.1	Architecture	19
5.2	Fragments	19
5.2.1	Benefits and Limitations	21
5.2.2	Fragments in WOMBAT	21
5.3	Lists	21

5.3.1	Benefits and Limitations	22
5.3.2	Lists in WOMBAT	24
5.4	Customize	24
5.4.1	Architecture of Customize	25
5.4.2	Buttons in Customize	28
5.5	Backend Library	32
5.5.1	TimerLib	32
5.5.2	DrawLib	32
6	Test	34
6.1	Test Design	34
6.2	Test Cases	38
6.3	Test Results	45
6.3.1	Reflections	46
6.4	Usability Test	46
6.4.1	Results and Observations	46
6.5	Acceptance Test	46
6.5.1	Results and Observations	47
III	Discussion	49
7	Reflections and Evaluations	51
7.1	Conclusion	51
7.2	Future Work	51
7.3	Conclusion	52
7.4	Future Work	52
7.4.1	Making a Handset Version	52
7.4.2	Better Graphics	52
7.4.3	New Design Features	52
IV	Appendix	55
7.5	Paper Prototypes	56
7.6	Sprint Burndown Charts and Backlogs	60
7.7	Acceptance Test Diary	66
Bibliography		67

Special Words

- Time Timer - a visual clock. Used by parents and teachers as a tool to visualize time for children.

Part I

Introduction

In this part there will be an introduction to the project, including background knowledge about the target platform, and knowledge about autism. There will be an analysis of the problem followed by a system definition of the whole multi project, and the specific group project.

CHAPTER 1

Analysis

Through meetings with Mette Als Andreasen, an educator at Birken, a special kindergarten for children with autism, we have learned a lot about children with autism, and the importance of having access to well-designed communication tools.

At Birken they often use hourglasses, and other kinds of timers, in different sizes and colors to visualize the progression of time to the children. The children will then associate the color and size of an hourglass with the time it represents, and specific timers are always used when they are performing specific activities, i.e. they always spend 30 minutes on eating lunch.

Mette Als Andreasen also explained how they use pictograms to communicate with the children. They have a scheme for the day, where all their daily activities are listed in pictograms, so the children can always go to their schemes and see what they are going to do next. Also activity instructions are listed with pictograms, i.e. in the bathroom there is a scheme showing how to wash hands.

The pictograms used at Birken comes from a licensed piece of software called *Boardmaker*[6]. To use the pictograms, the educators have to choose and edit them on the computer, print them out, cut them out in small squares, and laminate them. After that they can be put to use either on the schemes or by showing them when needed.

In general the guardians need a lot of different tools all the time, and the tools are not very practical to transport. Therefore it would be practical to

have a digital version of the timers and pictograms, so they would only have to bring one tablet, where all the needed tools are available. This leads to the system definition of this subproject, which is found in the next section.

1.1 System Definition

The application we are developing is targeted for android tablets running Android 3.2. The use space is institutions and homes of autistic children.

The application is meant as a tool, such that parents and educators can visualize time in a way customized for each child by changing color schemes, symbols, forms, and save this information in profiles stored on a server. The visualization is formed as a full-screen timer, which can be customized to be shown as an hour glass or a stop watch.

Furthermore the guardians should be able to add pictograms to the timer view, to show them what they are going to do while the time is running, and what they are going to do when the time has run out.

Tail

Part II

Development

In this part we start with a section about project management, where we expand on the used development method and versioning. After that we describe the design, development process, implementation, and test of the WOMBAT application.

CHAPTER 2

Project Management - on a Multi-Project Level

This project is a multi-group project, and five groups of three to four people are working together to develop one combined product. For this to work out, it is necessary to have a development method which suits this kind of multi-group development. Through the course *Software Engineering*, a part of our study regulation, we learned about SCRUM, and we chose to adopt this technique.

Section 2.1 is based on a mini-project written in the course *Software Engineering*, and it explains the modifications we have applied to the SCRUM method.

2.1 Modification of SCRUM

Even if we decided that SCRUM would best suit our situation, we had to modify the method to make it fit better. The modifications are as follows:

- Daily SCRUM is substituted with weekly SCRUM.
- We have no SCRUM master. The group is together responsible for the SCRUM master's tasks.
- Extensive documentation is done, since the bachelor project next year is going to be based on the results of this project.

- Instead of a project owner we have a democracy between all groups. This is because everyone has to learn from this project and therefore everyone has to be engaged all phases of the development.
- We are limited in time, so we are probably only going to do three 2-week sprints in total.

Arguments for Chosen Method Since this project requires the entire android group to be split into smaller groups of 3-4 people, we are going to use a development method which supports small groups working together. It is important that we develop prototypes, showing the project concepts in progress, to make sure we reach a usable final product. Since everyone involved in the project is novices in the field, students being new to working with autistic children and educators being new to working with software developing, an agile method would be suitable, as changes in demands to the product might occur often.

Agile development meets that the software is developed as part of a learning process.

2.2 Versioning

For versioning the code and documentation, we use a SVN-server at *Google Code*[5].

There several folders on the server, which help the groups to keep track of working code, reports, etc.. The structure is as follows:

- *trunk* - contains iteration releases of the code.
- *branches* - contains code under development.
- *tags* - contains the newest final release of the code.
- *common_report* - contains the common part of the report.
- *Reports* - contains all group reports.
- *wiki* - contains summaries from all group- and supervisor meetings, and guides/agreements.

CHAPTER 3

Design

The design has changed throughout the project. In this chapter we present the vision with the timer application, and some of the tools we have used when designing and re-designing.

3.1 Vision

Since the launcher had both a child-mode and a guardian-mode when we came with the initial design, this vision relies on the vision of the launcher project.

The initial idea of the WOMBAT system was a three-application tool to help educators illustrate time for the children they work with. One part is the timer application as it is implemented, from which it is possible to run customized timers, such as an hourglass or a digital watch. This part should be available only from the guardian-mode.

The second part of the system is a timer overlay, which is launched when other applications, for example game applications, are launched through the GIRAF launcher. When such applications are launched through guardian mode, the user should be prompted to select if there is a time limit on the application, and what the limit should be. If a time limit is chosen, the given application, i.e. a game, is run with the timer overlay showing a custom timer with the time left (see figure 3.1).



Figure 3.1: Example of how we imagined how the timer overlay would look. Here there is an overlay on the Angry Birds game with about 40 minutes left.

If the launcher is in child-mode, and the child opens some application, the timer overlay is run according to settings chosen in the third and last part of the WOMBAT system: the settings application. The overlay can be used if a child is only allowed to play a game for 30 minutes a day, then the overlay can be customized to show the child how long time is left of the allowed time. When the time is run out, the application is automatically closed.

The settings application is only available from the guardian-mode in the launcher, and from this application it is possible to customize which applications should be run with the timer overlay. Also it is possible to define how the overlay for every child should look like, and what should happen when the time limit has been reached. Also it is also possible to set constraints for the applications, such as time constraints, i.e. certain applications can only be opened in a different timespan on the day, or when a specific application has been run for 30 minutes straight, the application is closed and cannot be opened before a certain "cooldown" time.

3.2 Use Cases

A few use cases for each part of the vision.

3.3 Prototyping

Before the actual development started, we made a few drawings of our ideas using a drawing application on the computer (see figure 3.2).

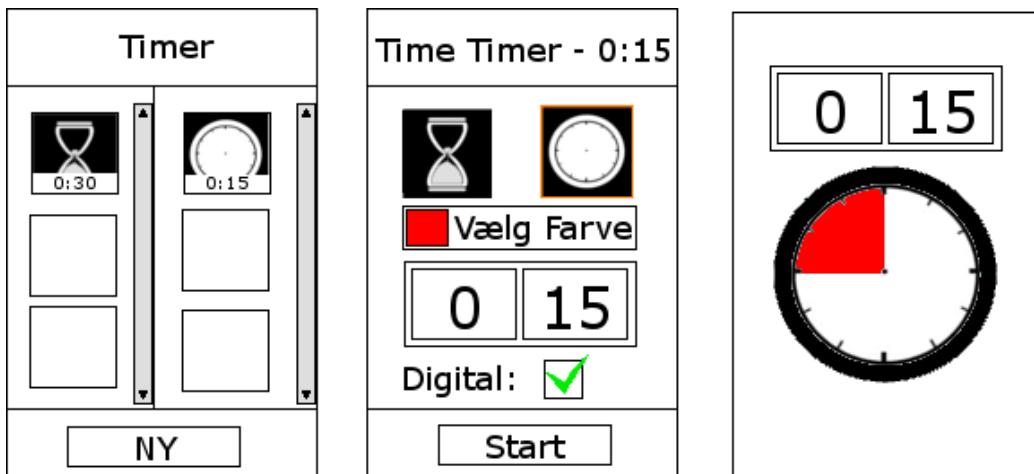
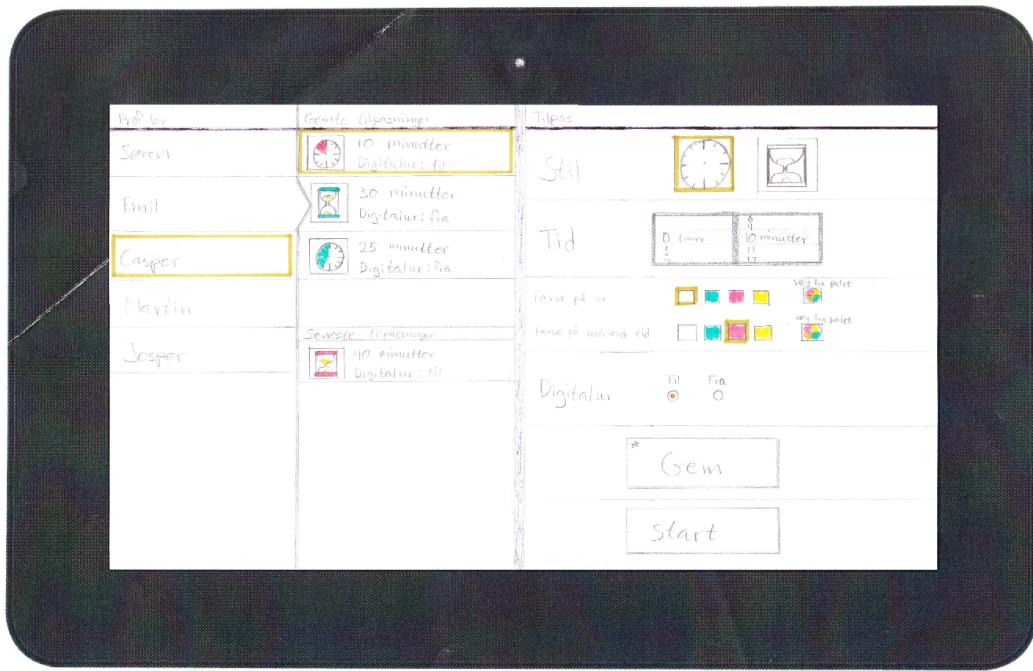


Figure 3.2: Drawing of the initial ideas of the timer application.

Paper Prototyping

Paper prototypes[1] has been used in the first iterations in the development process. The initial idea of the system design was drawn on paper, so that our contact person, Mette als Andreasen, could give us some feedback on the design, before we started programming. Furthermore it made it easier for us to program, when there was a specific design to work towards. In figure 3.3 is a paper prototype of the menu in the timer application, and the rest of the prototypes can be found in appendix 7.5.



* Pop-up menu, hvor man vælger hvor der skal gemmes.
For mulighed for at oprette tidlige gentle.

Figure 3.3: Scan of a paper prototype of the menu in the timer application.

Paper prototypes are produced quickly, and they capture early design ideas.

3.3.1 Metaphors

To enhance usability and learnability, we have used metaphors[1] on the buttons in the application. On the "Attach"-button, used to attach a second timer or one or two pictograms to the main timer, we have placed a paper-clip, which is known from the attach function in other programs, for example Microsoft Outlook Express. Furthermore we have used metaphors on the "Start Timer"-button, which looks like the "Play"-button known from various media players, and the "Save" and "Save As" buttons have a floppy disc icon, which is known from the save button in various word processing programs, for example Microsoft Office Word¹. In figure 3.4 examples of the implemented metaphors are shown along with screen-shots of other systems they are implemented in.

¹Non-free word processor developed by Microsoft

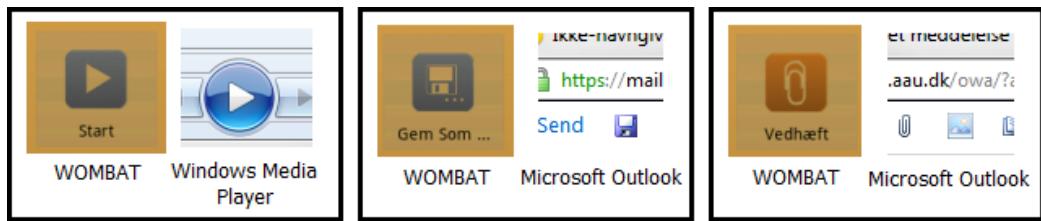


Figure 3.4: Metaphors implemented in WOMBAT vs. original implementation.

CHAPTER 4

Development Process

As stated in chapter 2, the development method used in this project is a modification of SCRUM, which means that the development evolve through sprints. Here is a description of the six development sprints we had, and in appendix 7.6 all sprint backlogs can be found.

4.1 Sprint Walk-through

Because it took some time for all groups to agree on a development method, we started making some design and talking to our contact person before the actual sprints started, so when the first sprint was started, we already had an idea of the functionalities we wanted to end up with.

Sprint 1: 19/03 - 23/03

In the first sprint, the main objective was to set up the android project in development environment, and get the development started. At this point we were expecting to end up implementing the whole vision (see section 3.1), and a natural place to start would be the timer. We had an idea of how the design should be, and we began to implement lists for holding children and configurations.

Sprint 2: 26/03 - 04/04

In the middle of this sprint the launcher group stated that they would only be able to make the guardian mode in the launcher in this semester, so we decided in the timer group, that the two last parts of the timer application, the overlay and the settings, would not be developed in this semester either, since there will be no need for them if there is no child mode in the launcher. We continued designing and developing, and in the end of this sprint, we had the design ready for implementation. We also started to make some OpenGL development¹, as we thought that OpenGL was the best way to implement the timers.

We had a meeting with the contact person, and she suggested that when the time has run out, it would be possible to show two custom pictograms on the screen. Also she suggested that two different timers could run in parallel on the screen, if a child was used to one specific timer, but was learning how to use another type of timer. We added these to suggestions to the product backlog.

Sprint 3: 10/04 - 19/04

We found out during this sprint, that it was more convenient to implement the timers using canvas and 2D drawings. The progress bar had been implemented in both OpenGL and on canvas, and the canvas version was the easiest to implement, and it was on fewer lines of code.

Sprint 4: 23/04 - 04/05

In this sprint we finished drawing the timers, and a lot of the most vital functionalities, such as loading and saving configurations, highlight on list items, and the "Done" screen, were implemented.

During this sprint we had a lot of contact with the admin group, because we had difficulties implementing functions to load and save data to the database. We also integrated the timer with the launcher, and did some testing on the interaction between them.

Sprint 5: 07/05 - 11/05

This sprint was used to do some refactoring of the code, to make it more readable and understandable. We also implemented the last things from the backlog, and did some polishing to the overall design. Furthermore we

¹Widely used 2D/3D graphics API

started making test design and test cases for the functionalities we want to test in the timer application.

Sprint 6: 14/05 - 18/05

In this sprint we implemented the last critical functionality. The only thing on the backlog we did not implement is the pictures in the profile list, because of missing functionality from the admin group. This part is only for showing the picture of the children on the profile list items, so it is not an important function.

Besides getting done with the development, we started with the acceptance test, by letting the contact person use the tablet with the timer application in everyday scenarios in the institution.

CHAPTER 5

Implementation

Nice...

5.1 Architecture

*UML diagram over biblioteker
Flow chart over WOMBAT livscyclus
Arkitektur diagram over WOMBAT
Kort beskrivelse af hvad en aktivitet er*

5.2 Fragments

In short a fragment in Android is functioning much like iFrames in HTML, meaning that a fragment is an embedded activity within another activity. According to Google the design philosophy of fragments are that they support a more dynamic and versatile design of applications on devices with larger screens, such as tablets. Google states that on these kind of devices there are more room to combine and interchange interface components, compared to a handset. An example of the use of multiple fragments on a large screen compared to activities on smaller handsets, is the Gmail application from Google:

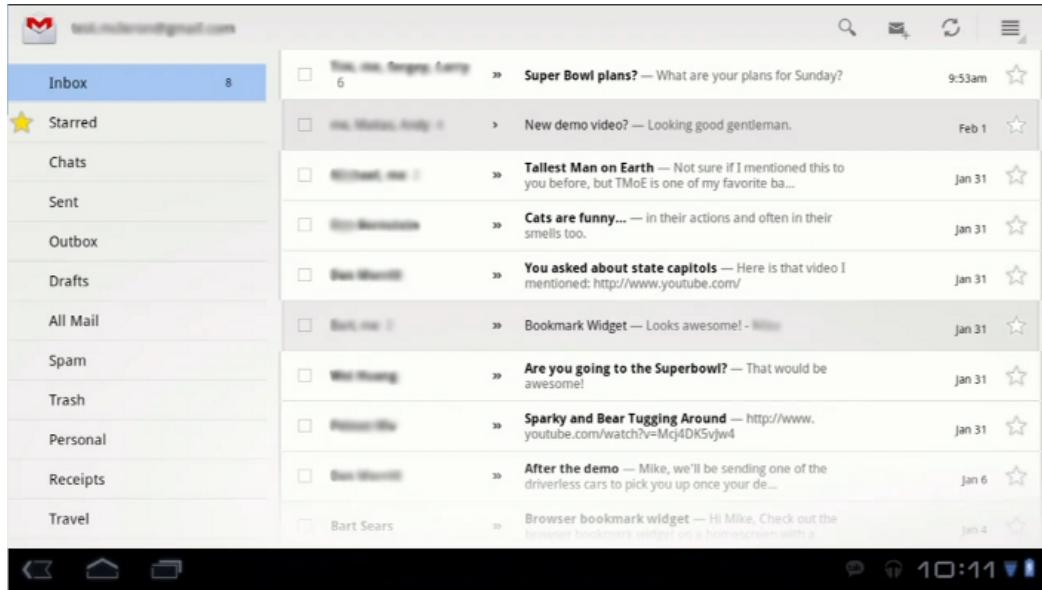


Figure 5.1: The Gmail application from Google on a tablet with fragment support.

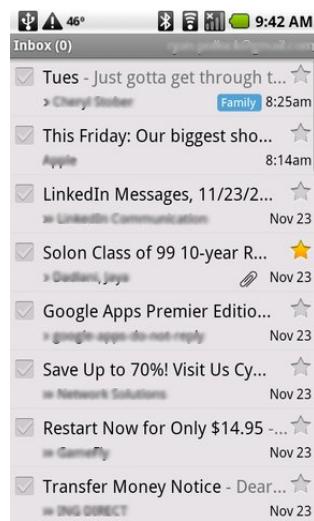


Figure 5.2: The Gmail application from Google on a handset without fragments

This and more information about fragments can be found at [4].

5.2.1 Benefits and Limitations

Fragments alone are not more powerful than activities and fragments can not exist without activities. When combining multiple fragments in one activity, the fragments are able to create dynamic interfaces which does not require the entire activity to be reloaded when changing an object on the screen.

The Gmail application utilizes this by having an overview of the email account on the left side of the device while showing specific mail content on the right side as seen in figure 5.1.

Fragments are built to be reused in multiple activities, which means that a single fragment can be used in several ways depending on the platform. The main activity only has to declare which fragments it holds in the layout.

Fragments were introduced the first time with the Android 3.0 platform (API version 11). But as of 3rd March 2011 Google has made the "Android Compatibility Package", a library which let Android 1.6 devices or newer support fragments [2]. Mobile tuts+, a website about tutorials for Android development, has tested the "Android Compatibility Package" and they successfully used fragments with the support library on the T-Mobile G1 (the first Android device) [3].

5.2.2 Fragments in WOMBAT

In WOMBAT we use fragments to give the user the ability to quickly switch between multiple children and configuration templates, when developing with fragments we are able to let each part of the screen update independently based on actions in the fragments. This means that we, much like the Gmail application, can keep an overview of all children and their personal configurations, while having a detailed configuration page open at the same time. Thus avoid requiring the user to switch screen to load previous defined configurations.

Having developed WOMBAT with fragments means that it is possible to create a handset version with few modifications, since it only requires the "Android Compatibility Package" library and some new activities to handle the fragments on a smaller screen.

5.3 Lists

In Android the philosophy behind lists is that the majority of all applications at some point will have to implement a list object. Therefore does Android

ship with a build in list view, that supports a variety of list. This build in list view can be modified to match any list, that the developer wishes to create, as shown in 5.3.



Figure 5.3: Examples of how ListViews can be modified.

5.3.1 Benefits and Limitations

The advantage of a list view is, that we can modify the list to whatever we want it to look like without having to create any complicated programming to make it look good. This is done by making an adapter with the layout that the items of the list is going to look like as in figure 5.4, which is the adapter we use to show the children list.

```

1 View v = list_item_view;
2     if (v == null) {
3         LayoutInflator li = (LayoutInflater) getContext().getSystemService(
4             Context.LAYOUT_INFLATER_SERVICE);
5         v = li.inflate(R.layout.profile_list, null);
6     }
7 // FIXME: Insert pictures from admin here
8 Child c = items.get(position);
9     if (c != null) {
10         ImageView iv = (ImageView) v.findViewById(R.id.profilePic);
11         TextView tv = (TextView) v.findViewById(R.id.profileName);
12
13         if (iv != null) {
14             iv.setImageResource(R.drawable.default_profile);
15         }
16         if (tv != null) {
17             if (c.name == "Last Used") {
18                 tv.setText(R.string.last_used);
19             } else if (c.name == "Predefined Profiles") {
20                 tv.setText(R.string.predefined);
21             } else {
22                 tv.setText(c.name);
23             }
24         }
25     }

```

Figure 5.4: Example of an adapter to a list view, the actual adapter used in the children list

Creating lists in adapters and adding adapters to list views is alot easier than making custom lists. Because these built in list views are very versatile and can be configured to do exactly the same as any other view in Android.

Since lists wont be able to do anything but show a list of items without an on click listener does the list view have an interface which works exactly like any other clickable item in android. The on click listener in the list view is as easy to implement as a button or similar clickable object.

5.3.2 Lists in WOMBAT

Figure 5.4 is the actual implementation of the child list in WOMBAT, the only difference from the child list adapter and the configurations adapter is that the timer configurations require another text field with a description of the timer. Which means that the configurations list only requires another layout to be constructed, which holds the extra text view.

The on click listeners of both lists, ensures that the next fragments and Guardian object, described in ??REF MANGER , is updated figure 5.5 is an example of the child fragment listview.

```
1  public void onListItemClick(ListView lv, View view, int
2      position, long id) {
3          // Update the fragments
4          SubProfileFragment detf = (SubProfileFragment)
5              getFragmentManager()
6                  .findFragmentById(R.id.subprofileFragment);
7          CustomizeFragment custF = (CustomizeFragment)
8              getFragmentManager().findFragmentById(R.id.
9                  customizeFragment);
10         custF.setDefaultProfile();
11
12         if (detf != null) {
13             // Marks the selected profile in the guard singleton
14             guard.profilePosition = position;
15             guard.publishList().get(position).select();
16             guard.profileID = guard.publishList().get(position).
17                 getProfileId();
18             detf.loadSubProfiles();
19         }
20     }
```

Figure 5.5: Example of the on click listener of the child fragment

5.4 Customize

The customize fragment is the heart of the application, this is where everything is generated and configured. The fragment can both create new and modify already existing timers, this is the key functionality of the customize fragment.

The timers that the guardians already use at the institutions comes in many different colors and timespans. Therefor is it possible to modify the look, time, and color of the individual timers in the customize fragment. Furthermore did the guardians state that a good feature would be if either pictograms or other timers were attachable to the timers.

Combined with the general idea of the application with a child and timer overview, is the requirements of the features in the customize fragment as follows:

- Change style of the timer
- Change the timespan of the timer
- Change the color of the timer and background
- Change the color of the timer to be changing gradiently
- Attach a pictogram or a timer to the timer
- Change the done picture of the timer
- Save the timer
- Start the timer

Det hører næsten mere til et design afsnit

5.4.1 Architecture of Customize

The customize fragment consists of four main elements, the first element is the style picker where the user can change the style of the timer. The second element is the time wheels where the user can change the timespan of the timer. The third is the color pickers where the user can change the colors of the time left, the frame, and the background of the timer. The last element is the advanced element where the modifications like attachment can be found. Figure 5.6 show how the fragment has been outlined.

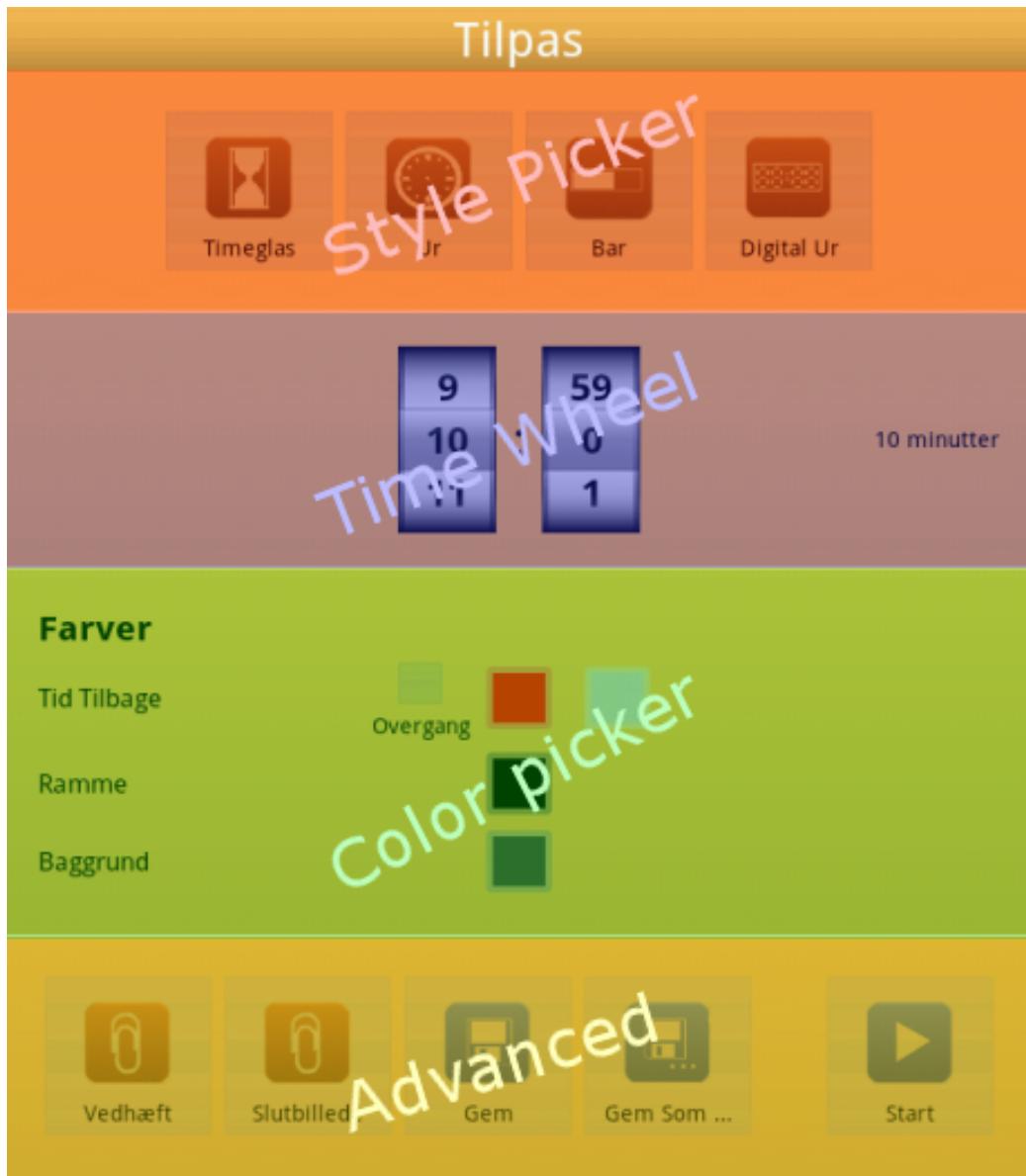


Figure 5.6: An outline of the elements of the customize fragment.

In the actual customize class has all buttons been initialized as global objects for quick reference. Since everyone on the WOMBAT project were new to Android programming when the structure of the class were established, is all functionality of the items in the customize fragment in this class. For further development would a refactoring of the entire class be advised such that all buttons in WOMBAT is handled like the WDialog.

To be able to get an overview of the items in the customize fragment has

all items been assigned a method which initializes the item e.g. the style chooser in figure 5.7. These methods are then referenced from the onCreate method in figure 5.8.

```
1 private void initStyleChoser() {
2     hourglassButton = (Button) getActivity().findViewById(
3         R.id.houglassButton);
4     hourglassButton.setOnClickListener(new OnClickListener() {
5
6         public void onClick(View v) {
7             selectStyle(formFactor.Hourglass);
8         }
9     });
10
11    timetimerButton = (Button) getActivity().findViewById(
12        R.id.timetimerButton);
13    timetimerButton.setOnClickListener(new OnClickListener() {
14
15        public void onClick(View v) {
16            selectStyle(formFactor.TimeTimer);
17        }
18    });
19
20    progressbarButton = (Button) getActivity().findViewById(
21        R.id.progressbarButton);
22    progressbarButton.setOnClickListener(new OnClickListener() {
23
24        public void onClick(View v) {
25            selectStyle(formFactor.ProgressBar);
26        }
27    });
28
29    digitalButton = (Button) getActivity().findViewById(R.id.
30        digitalButton);
31    digitalButton.setOnClickListener(new OnClickListener() {
32
33        public void onClick(View v) {
34            selectStyle(formFactor.DigitalClock);
35        }
36    });
37}
```

Figure 5.7: The style choser initialization method, which utilizes the selectStyle that changes the style of the timer and highlights the button.

```

1  public void onActivityCreated(Bundle savedInstanceState) {
2      super.onActivityCreated(savedInstanceState);
3      currSubP = new SubProfile("", "", 0xff3D3D3D, 0xffFF0000, 0
4          xffB8B8B8,
5          0xff000000, 600, false);
6      currSubP.save = false;
7      currSubP.saveAs = false;
8
9      //***** TIME CHOSER *****/
10     initStyleChoser();
11
12     //***** TIMEPICKER *****/
13     initTimePicker();
14
15     //***** COLORPICKER *****/
16     initColorButtons();
17
18     //***** ATTACHMENT PICKER *****/
19     initAttachmentButton();
20
21     //***** BOTTOM MENU *****/
22     initBottomMenu();
23 }

```

Figure 5.8: The onCreate method, which calls the button initializers in the same order as they are shown in the layout.

For successors on the WOMBAT project does Eclipse have a keyboard shortcut (F3) which jumps to the method your cursor is at, that can be used to quickly find e.g. the color buttons in the customize class.

5.4.2 Buttons in Customize

There is roughly four kinds of buttons in the customize fragment:

- Start, Save, and style buttons
- Time picker wheels
- Color picker
- Attachment and Done picture button

Start Button

The style, save, start, and attachment buttons are ordinary Android buttons with a picture attached to the top side. The differences of these buttons is the onclick event handler which is slightly different from button to button, figure 5.9 is the source code of the start button.

```
private void initStartButton() {
    2   startButton = (Button) getActivity().findViewById(
        R.id.customize_start_button);
    4   Drawable d;
    5   if (currSubP.saveAs) {
    6       d = getResources().getDrawable(R.drawable.thumbnail_start);
    7       startButton.setOnClickListener(new OnClickListener() {
    8           public void onClick(View v) {
    9               currSubP.addLastUsed(preSubP);
    10              guard.saveGuardian(currSubP);
    11              currSubP.select();
    12              Intent i = new Intent(
    13                  getActivity().getApplicationContext(),
    14                  DrawLibActivity.class);
    15              startActivity(i);
    16          }
    17      });
    18  } else {
    19      ...
    20  }
    21 }
    22
    23 startButton
    24 .setCompoundDrawablesWithIntrinsicBounds(null, d, null, null);
    25 }
```

Figure 5.9: The source code of the start button, which sets the top image of the button to the drawable `thumbnail_start_gray`

Time Picker Wheels

The time picker wheels is a wheel widget created by yuri.kanivets@gmail.com [8], these wheels can be customized to mimic the time pickers from the iPhone. All functionality of the wheel widget is handled by the widget itself and only requires the widget to be imported to Eclipse and added as a

library, which is a built in feature in the Android development environment. To avoid a situation where the user would try to set the time to more than 60 minutes, did we implement a functionality which sets the seconds wheel to zero whenever the minutes wheel is set to 60, this can be seen in figure 5.10.

```

1  private int previousMins;
2  private int previousSecs;
3  private void initTimePicker() {
4      /* Create minute Wheel */
5      mins = (WheelView) getActivity().findViewById(R.id.minPicker);
6      mins.setViewAdapter(new NumericWheelAdapter(getActivity()
7          .getApplicationContext(), 0, 60));
8      mins.setCyclic(true);
9
10     /* Add on change listeners for both wheels */
11     mins.addChangingListener(new OnWheelChangedListener() {
12         public void onChanged(WheelView wheel, int oldValue, int
13             newValue) {
14             updateTime(mins.getCurrentItem(), secs.getCurrentItem());
15
16             if (mins.getCurrentItem() == 60) {
17                 previousMins = 60;
18                 previousSecs = secs.getCurrentItem();
19
20                 secs.setCurrentItem(0);
21                 secs.setViewAdapter(new NumericWheelAdapter(getActivity()
22                     .getApplicationContext(), 0, 0));
23                 secs.setCyclic(false);
24             } else if (previousMins == 60) {
25                 secs.setViewAdapter(new NumericWheelAdapter(getActivity()
26                     .getApplicationContext(), 0, 60));
27
28                 secs.setCurrentItem(previousSecs);
29                 secs.setCyclic(true);
30                 previousMins = 0;
31             }
32         }
33     });
34 }
```

Figure 5.10: The time picker wheels, with the functionality which ensures the time is never set to more than 60 minutes

Color Picker

Just like the time picker wheels did we utilize the widget functionality in Android to implement a widget called AmbilWarna (means Take Color in Indonesian), created by Yuku Sugianto [7], to handle the color picker. The color picker widget is a custom dialog which returns the color picked by the user in the widget, the implementation of the color picker widget can be found in figure 5.11.

```
1 colorGradientButton2 = (Button) getActivity().findViewById(      R.id.gradientButton_2);  
2 setColor(colorGradientButton2.getBackground(), currSubP.  
3   timeSpentColor);  
4 colorGradientButton2.setOnClickListener(new OnClickListener() {  
5   public void onClick(View v) {  
6     AmbilWarnaDialog dialog = new AmbilWarnaDialog(getActivity()  
7       ,  
8       currSubP.timeSpentColor, new OnAmbilWarnaListener() {  
9         public void onCancel(AmbilWarnaDialog dialog) {  
10        }  
11  
12         public void onOk(AmbilWarnaDialog dialog, int color) {  
13           currSubP.timeSpentColor = color;  
14           setColor(colorGradientButton2.getBackground(),  
15             currSubP.timeSpentColor);  
16         }  
17       });  
18       dialog.show();  
19     }  
});
```

Figure 5.11: The color picker widget implement on the second "Time Left" button

Attachment Button

The attachment buttons are implemented like the other buttons, but they make use of a custom dialog in WOMBAT. We created this dialog, because the standard dialogs in Android were very different from the rest of the WOMBAT design. Also were we unable to define exactly what kind of buttons we wanted in our dialogs.

Therefor did we implement the custom dialog which would match the rest of our WOMBAT design and give the ability to control exactly what the but-

tons on the dialog should look like and what they should do when clicked. Figure 5.12 is an example of how a dialog could be specified.

```
1 final WDialog attachment1 = new WDialog(getActivity(),  
2     R.string.attachment_dialog_description);  
3  
4 ModeAdapter adapter = new ModeAdapter(getActivity(),  
5     android.R.layout.simple_list_item_1, mode);  
6  
7 attachment1.setAdapter(adapter);  
8  
9 attachment1.addButton(R.string.cancel, 1,  
10    new OnClickListener() {  
11        public void onClick(View arg0) {  
12            attachment1.cancel();  
13        }  
14    });  
15 };
```

Figure 5.12: Initialization of the dialog, that appears when the attachment button is clicked, the onItemClickListener is not shown here because of lack of space on the page.

5.5 Backend Library

Introduktion til hvorfor det er smart at bruge biblioteker frem for at udvikle på ét workspace

(Man kan nemmere teste på et bibliotek hvor man kan lave en test app, frem for at være afhængig af at "main" appen virker)

5.5.1 TimerLib

*Beskrivelse af timerlib, hvad indeholder timerlib, services
Beskrivelse af implementationen af timerlib med kode eksempler*

5.5.2 DrawLib

*Beskrivelse af DrawLib, hvad indeholder DrawLib, services
Beskrivelse af implementationen af DrawLib med kode eksempler
Hvad er fordele ulemper for canvas frem for openGL*

*Introduktion til tegninger med canvas i Android
Beskrivelse af implementationen af timere og hvordan bevægelser er udregnet
Optimering af timere og de problemer vi løb ind i under vejs (canvas til bitmap
og systemtimer)*

Canvas and OpenGL comparision

CHAPTER 6

Test

The WOMBAT application is tested through dynamic black box testing, which means that we run the application, and test the functionality through the user interface, without viewing the code. We have made test design and test cases for some of the most important functionality, so that we could outsource the test to one of the other project groups, if they had time to help us. This was not the case, so we did the testing ourselves.

6.1 Test Design

The test designs are split into four schemes, which are listed in this section.

Identifier Save and load.

Feature Saving and loading configurations.

Approach NB! Check if the configuration has changed in the database by closing the application and resetting the memory (RAM).

- Create a configuration in three ways:
 - Click "New Template" and click "Save As".

- Edit a current configuration and click "Save As".
- Check if it is possible to "Save As" a configuration in "Predefined" or "Last Used".
- Check if loaded settings are the same as the when they were saved.
- Edit and save (by clicking "Save") each of the configuration:
 - Check if it is possible to save configurations in "Predefined" or "Last Used".
 - Check if they have the settings they were saved with.
 - Check if there is any duplicates of any of the configurations.
 - Check if any other configuration was changed while editing.

Test case ID

1. Check save as functionality - *saveAs#1*
2. Check save functionality - *save#1*

Pass/fail criteria Pass

- It is possible to create a new configuration by clicking "New Template" and "Save As".
- It is possible to make a new configuration by clicking "Save As" on any configuration.
- No profiles in "Last Used" or "Predefined" is editable.
- It is not possible to save new profiles to "Last Used" or "Predefined".

Fail

- If any of the above does not hold.
- When saving with "Save" the configuration is being duplicated.
- When saving another configuration is being altered.

Identifier Last used is updated correct

Feature When a timer has been used, it should lie in the top of the list of last used configurations.

Approach

1. Start any timer
 - Check if the timer has been added to the last used list
2. Repeat step [1] 7 times with different timers
3. Start any of the timers in "Last Used" and check if it is being moved to the top of the list.

Test case ID

1. Check "last used" functionality - *checkLastUsed#1*

Pass/fail criteria Pass

- Every time a timer used, it is saved on the top of the "Last Used" list.

Fail

- If, in any case, a timer is not saved in the "Last Used" list after it has been used.
-

Identifier Highlight is working

Feature When choosing a child or a configuration this should be highlighted, and when Wombat is started from the GIRAf launcher a child is chosen, this should be highlighted.

Approach

1. Test if children and configurations is being highlighted when they are chosen.
2. Test if the child chosen in the GIRAf launcher is being highlighted in Wombat.
3. Test if the child chosen is still highlighted after saving.

NOTE! An element in the child list is chosen if there is configurations in the configurations list, and an element in the configurations list is chosen if there is loaded a configuration in the edit screen.

Test case ID

1. Check if list elements is highlighted when clicked - *hOnClick#1*
2. Check if child is still highlighted after save - *stillHAfterSave#1*
3. Check if the right child is highlighted after launch through the GIRAf launcher - *hChildOnLaunch#1*

Pass/fail criteria Pass

- If list elements are always highlighted when selected.

Fail

- If a child is selected and it is not highlighted.
- If a configuration is selected and it is not highlighted.
- If nothing is highlighted when starting Wombat from the GIRAf Launcher

Identifier Deviation in time on done activity

Feature When the timer has run out, the "Done" screen will appear.

Approach

1. Run a timer with any timespan, and wait for the "Done" screen to appear
 - Use an independent stopwatch to verify the time it takes to show the "Done" screen.
 - Verify that there is no more than a 2 second deviation in time, from the time has run out until the "Done" screen appears.
2. Repeat step [1] 3 times with different timespans.
3. Do step [2] again with any timer with a "Digital Watch" attached.
4. Start a timer with any timespan, click the "back" button, and verify that the "Done" screen do not show up randomly.

Test case ID

1. Check if timer matches real-life time - *checkTimerTime#1*
2. Check if the done screen appears when it should - *checkDoneFunc#1*

Pass/fail criteria Pass

- If the "Done" screen appears no more than two seconds after the time has run out.
- If any timer is not deviating more than two seconds from real-world time.

Fail

- The "Done" screen appears even though the timer is not running anymore or the timer is not finished.
-

6.2 Test Cases

Identifier *saveAs#1*

Test item Functionality to save customized timers in specific lists.

Input spec.

1. Click "New Template", choose a timer type, click "Save As", and choose name and location. Check if the profile was saved in the chosen location and with the chosen name.
2. Choose any configuration, edit the settings, click "Save As", and choose name and location. Check if the profile was saved in the chosen location and with the chosen name.
3. Create a new configuration with random settings and use "Save As" to save it. Go to the saved configuration, and check if the settings has changed since the save.

4. Choose an existing configuration or create a new one, and do step 1. with "Predefined" and "Last Used" as save locations.
5. Do step 1-3 again, but clear the tablet memory before the correctness is checked.

Output spec.

1. Configurations is saved in the chosen locations, unless the chosen locations is "Predefined" or "Last Used".
2. Configurations is saved with the chosen name.
3. Configurations is saved with the chosen settings.

environmental needs

- Tablet running Android 3.2.
 - Timer application installed.
 - OasisLocalDatabase installed.
 - One staff member to perform the test.
-

Identifier *save#1*

Test item Functionality to save customized or predefined timers in the highlighted child, without having to choose name and location.

Input spec.

1. Check if it is possible to save configurations in "Predefined" or "Last Used" by choosing one of the configurations in each list, edit some settings, and press "Save".
2. Select a child, edit the settings, and press "Save". Check if the chosen settings were saved.
3. Select a child, select a configuration among the child's configurations, edit the settings, and press "Save". Check if the chosen settings were saved in the same configuration.

4. Select a child, edit the settings, and press "Save" two times and see if two identical configurations are saved on the given child.
5. Highlight another configuration than the one you have just saved, then highlight the one you just saved and press "Save". Check if there is now saved a duplicate of the first saved configuration.
6. Do step 2-3 again and check if any other configuration was changed during the saving process.

Output spec.

1. Configurations is saved in the highlighted child.
2. When "Predefined" or "Last Used" is highlighted, nothing is saved when the "Save" is pressed.
3. New configurations are saved with the chosen settings.
4. When selecting and saving existing configurations, they are updated with the edited settings.

environmental needs

- Tablet running android 3.2.
 - Timer application installed.
 - OasisLocalDatabase installed.
 - One staff member to perform the test.
-

Identifier *checkLastUsed#1*

Test item Functionality to save timers into the "Last Used" list every any timer has been run.

Input spec.

1. Run 3 different timers, and see if they were saved on top if the "Last Used" list.
2. Repeat step 1, but clear the tablet memory before "Last Used" is inspected.

Output spec.

1. Whenever a timer has been used, it is saved on top of the "Last Used" list.

environmental needs

- Tablet running android 3.2.
 - Timer application installed.
 - OasisLocalDatabase installed.
 - One staff member to perform the test.
-

Identifier *hOnClick#1*

Test item Functionality to highlight list items when they are clicked.

Input spec.

1. Select three different list items in both the child list and the configuration list and see if they stay highlighted.

Output spec.

1. When a list item is selected, it is highlighted, and it stays highlighted until another list item is selected.

environmental needs

- Tablet running android 3.2.
- Timer application installed.
- One staff member to perform the test.

Special procedural requirements

- The configurations on every child will always be visible when a child list has been selected. Therefore, make sure that the highlighted child has at least one configuration before testing.
 - There is no element in the configuration list if no element in the child list has been selected.
-

Identifier *stillHAAfterSave#1*

Test item Functionality to highlight list items after a save procedure.

Input spec.

1. Select a child and a configuration, edit the settings for the configuration, and click "Save". See if the selected list items stay highlighted after it has been updated.

Output spec.

1. When a child and configuration is selected, and the settings for that configuration is changed and saved, the child list item and configuration list item is still highlighted.

environmental needs

- Tablet running android 3.2.
- Timer application installed.
- One staff member to perform the test.

Special procedural requirements

- The configurations on every child will always be visible when a child list has been selected. Therefore, make sure that the highlighted child has at least one configuration in the list before testing.
- There is no element in the configuration list if no element in the child list has been selected.
- When a configuration is selected, the settings for that configuration is always shown set in the "Customize" menu.

Intercase dependencies *save#1*

Identifier *hChildOnLaunch#1*

Test item Functionality to highlight list items according to the chosen child when the application is launched through the GIRAF launcher.

Input spec.

1. Start the GIRAF launcher and open the timer application.
2. Select a child and note the name of the child.

Output spec.

1. The child selected in the GIRAF launcher is highlighted and the configurations belonging to this child is loaded.

environmental needs

- Tablet running android 3.2.
- Timer application installed.
- GIRAF launcher installed.
- One staff member to perform the test.

Special procedural requirements

- The configurations on every child will always be visible when a child list has been selected. Therefore, make sure that the highlighted child has at least one configuration before testing.
 - There is no element in the configuration list if no element in the child list has been selected.
-

Identifier *checkTimerTime#1*

Test item Functionality which draws and updates the timer according to the time left and ensures the timer ends when the time is up.

Input spec.

1. Run four different timer styles with a static timespan (fx 20 minutes).
2. Each time a timer is started, start a precise independent stopwatch.
3. When the timer reaches zero stop the independent stopwatch.

Output spec.

1. The stopwatch must deviate no more than two seconds from the time selected in **input spec.** step [1].

environmental needs

- Tablet running android 3.2.
 - Timer application installed.
 - Stopwatch.
 - One staff member to perform the test.
-

Identifier *checkDoneFunc#1*

Test item The "Done" screen appearing when the time has run out.

Input spec.

1. Start a timer at any timespan and let the time run out. See if the "Done" screen appears within two seconds after the time has run out.
2. Start a timer at any timespan and click the "back" button, and wait at least the amount of time the timer would have run, to verify that the "Done" screen do not show up anyways, if the timer has been interrupted.

Output spec.

1. When a timer has been run, and not interrupted, the "Done" screen appears about two seconds after the time has run out.
2. The "Done" screen is only shown if the timer is not interrupted, and the time has run out.

environmental needs

- Tablet running android 3.2.
- Timer application installed.
- Stopwatch.
- One staff member to perform the test.

Intercase dependencies Test case: *checkTimerTime#1*

6.3 Test Results

Test Case ID	Pass	Fail	
<i>saveAs#1</i>	X		
<i>save#1</i>	X		
<i>checkLastUsed#1</i>		X	When tablet memory is cleared, the last used timers disappears.
<i>hOnClick#1</i>	X		
<i>stillHAfterSave#1</i>	X		
<i>hChildOnLaunch#1</i>	X		
<i>checkTimerTime#1</i>	X		
<i>checkDoneFunc#1</i>	X		

Table 6.1: Test results of the black box testing of the WOMBAT application.

6.3.1 Reflections

We did not outsource the test, so we did the testing ourselves. This is not the best way to do testing, because we wrote the program, and the test design and test cases, and thereby have a deeper knowledge about the features we want to test, and therefore we may have done things different than persons with no knowledge about the application would have done.

The only test that did not pass was known beforehand (*checkLastUsed#1*). We had trouble implementing the "Last Used" together with the features of the database developed by the admin-group. This resulted in sporadic system failure, so we decided to only use the tablet memory to save the last used timers, and save the correct implementation of this feature to future development.

During the test we stumbled upon a few vague formulations, which could have lead to misunderstandings and different testing method, if the tests had been outsourced. We learned that it is a good idea to run the test ourselves at first, so that we could correct any formulation errors we found, and then the test could be outsourced. This would require even more time, and is not likely to be done in large scale projects, but could have been done in our case.

6.4 Usability Test

6.4.1 Results and Observations

Test resultater og observationer

Tabel over hvilke funktionaliteter blev der fundet fejl ved og var det kritisk eller kosmetisk

6.5 Acceptance Test

The purpose of an acceptance test[1] is to test if the system fits into the context it is designed for. In this case the main objectives is to find out if it feels natural for the educators to use the timer application instead of their regular hourglasses and time timers, and to find out if the children understand the digital version of the different timers.

To test for acceptability, the system needs to be tested in the context it is developed for, and therefore the contact person borrowed the tablet with

the timer application for a few days, so that she could use the system in real life scenarios.

6.5.1 Results and Observations

The contact person wrote a diary, to keep track of her experiences with the application. The contact person used the timer application herself, and she let some of the other educators try it as well. The diary can be found in appendix 7.7.

They wrote in the diary that the children understood the meaning of both the timers and the pictograms shown on the tablet. One of the children was more interested in watching the digital timer count down than in the activity he was meant to do, but beyond that there were no problems understanding the system.

Beside the diary, we can analyze on the timers they have used in the acceptance test by looking in the "Last Used" list (see appendix 7.7).

We can see in the list that they have used different timers with different "Done"-pictures and times, only one out of eight timers had a slightly different color. This could indicate that changing the colors on the timers is not very intuitive, or they do not need the functionality anyway.

Tail

Part III

Discussion

Head

CHAPTER 7

Reflections and Evaluations

*Hvordan har brugen af pair programming og refactoring fungeret
Hvordan har SCRUM fungeret som projekt styring*

7.1 Conclusion

Konklusionen paa projektet

7.2 Future Work

*Port to handset
OpenGL timers
Remake of design (drawer with profiles etc.)
Remake gradient Button
Implementation af vision om overlay*

7.3 Conclusion

7.4 Future Work

In this section we will list some ideas for further development on the WOMBAT application.

7.4.1 Making a Handset Version

It is, at the moment, not possible to install and run the WOMBAT application on an Android handset, but there are several reasons for considering making a handset version of the application aswell.

7.4.2 Better Graphics

7.4.3 New Design Features

Tail

Part IV

Appendix

Appendix

7.5 Paper Prototypes

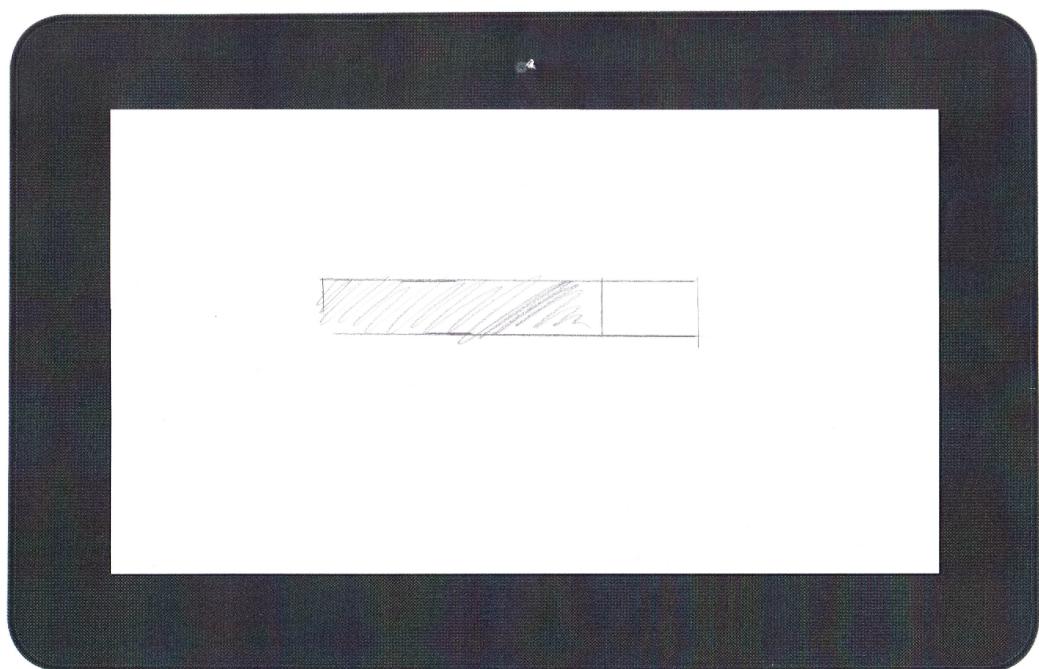


Figure 7.1: Scan of a paper prototype of a single progress bar timer.

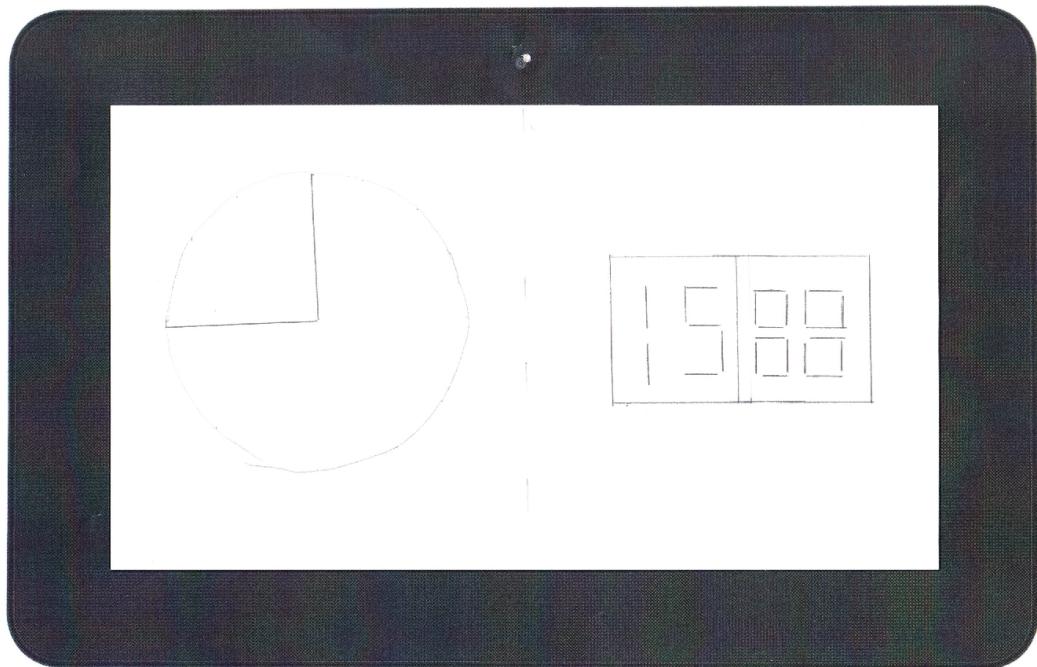


Figure 7.2: Scan of a paper prototype of a double timer.

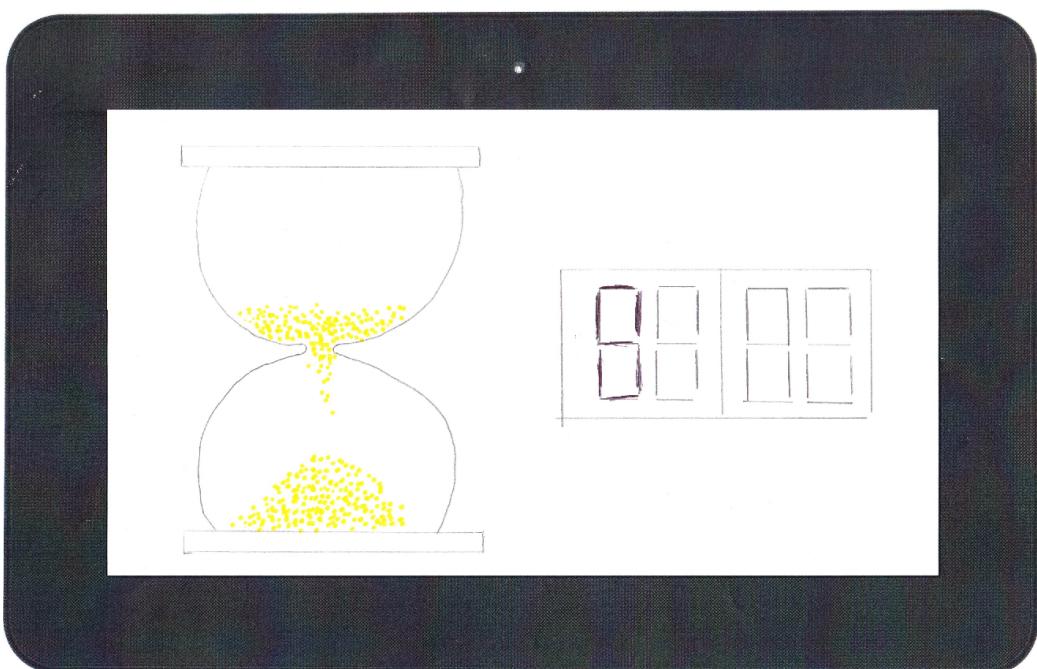


Figure 7.3: Scan of a paper prototype of another double timer.

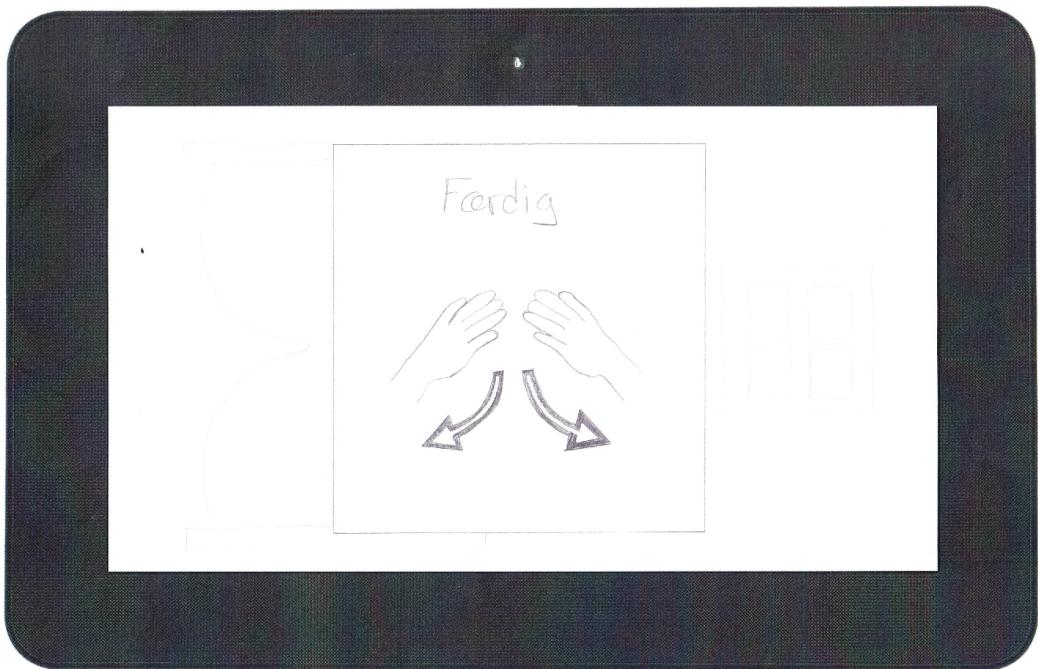
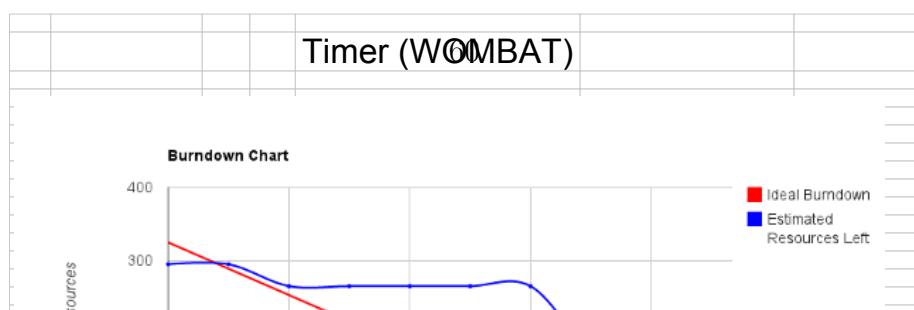
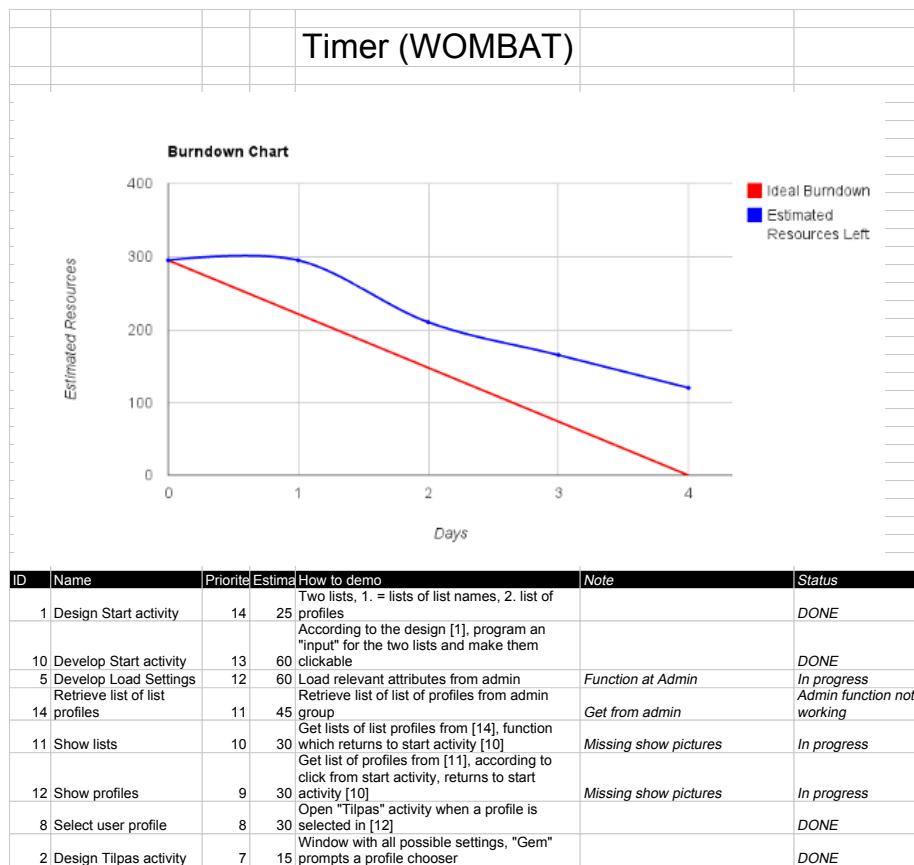
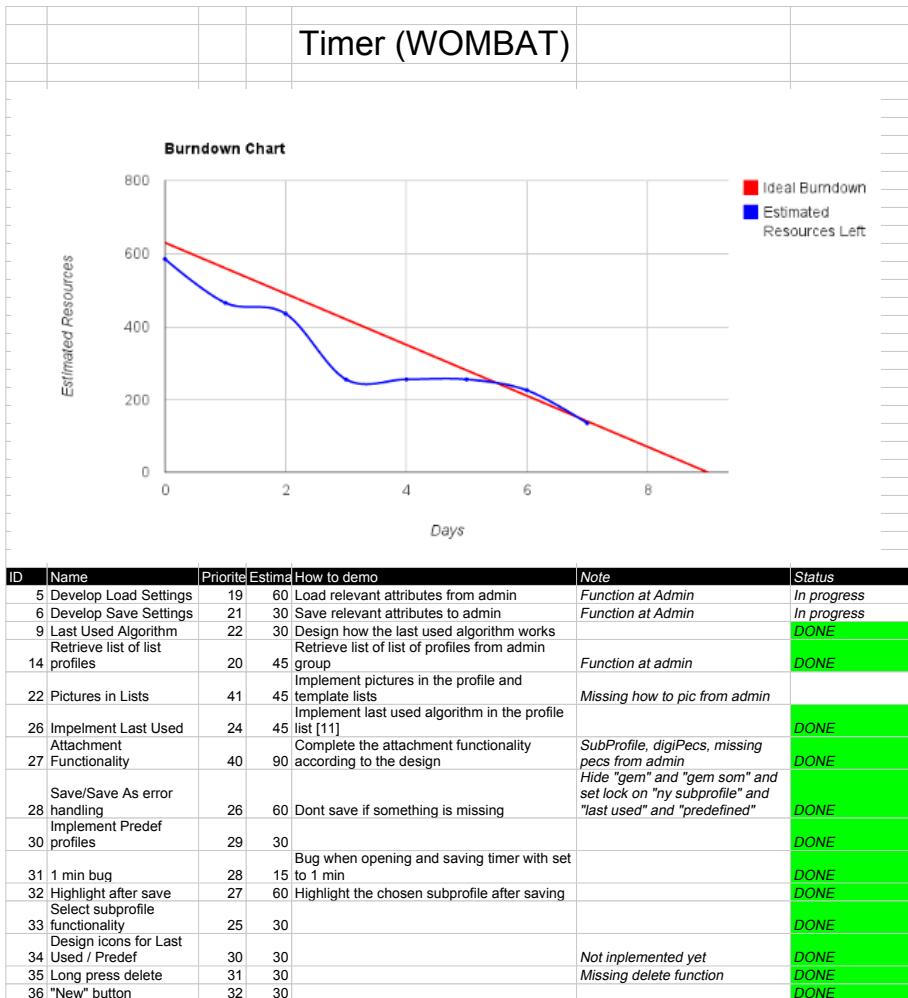
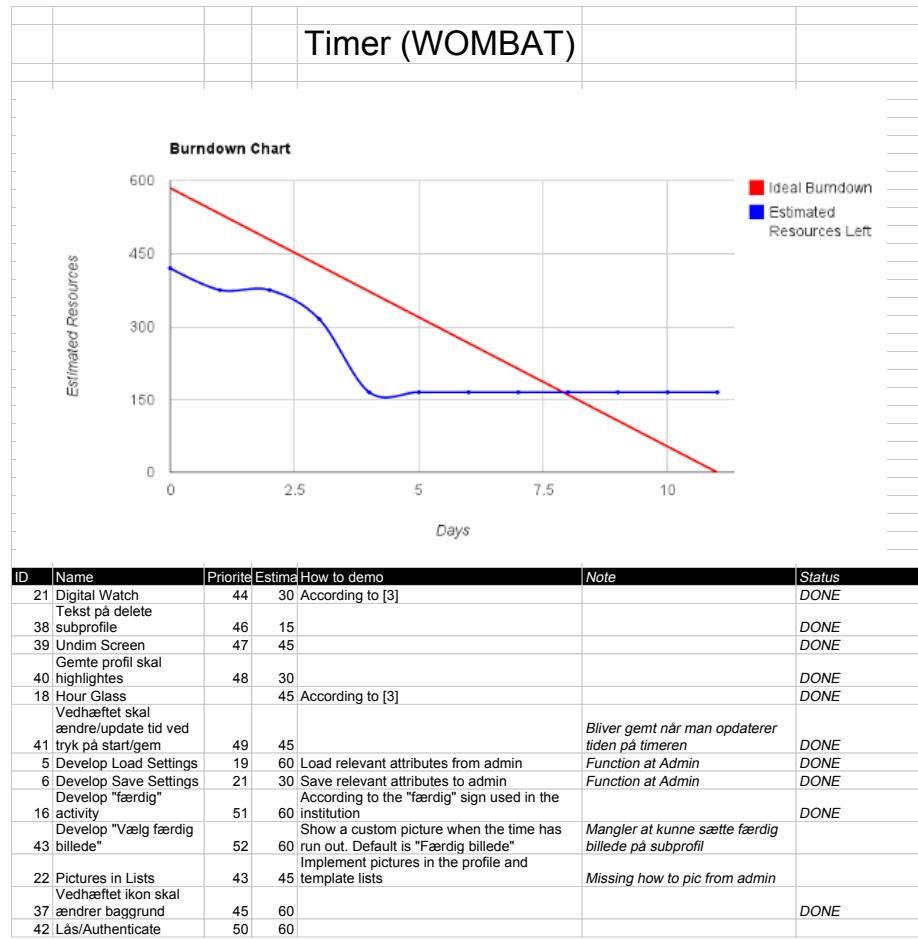


Figure 7.4: Scan of a paper prototype of the "Done" screen shown when the time has run out.

7.6 Sprint Burndown Charts and Backlogs











7.7 Acceptance Test Diary

Dato: 16/5 - 12

Mads

Virkede godt	Virkede knapt så godt/slet ikke
Var klart genkendelig for barnet (sandur)	slutbilledet var for hurtigt vek. Måske kunne det blive indtil jeg bestemmer andet.

Dato: 18/5 - 12



Figure 7.5: Scan of a paper prototype of the "Done" screen shown when the time has run out.

Bibliography

- [1] David Benyon. Designing interactive systems. Book, ? 2010. Pages: 88, 187, 203-207.
- [2] Tim Bray. Fragments for all. <http://android-developers.blogspot.com/2011/03/fragments-for-all.html>, 2011. Last visit: 15-05-2012.
- [3] Shane Conder and Lauren Darcey. Android compatibility: Working with fragments. <http://mobile.tutsplus.com/tutorials/android/android-compatibility-working-with-fragments/>, 2011. Last visit: 15-05-2012.
- [4] Google. Fragments | android developers. <http://developer.android.com/guide/topics/fundamentals/fragments.html>, 2012. Last visit: 15-05-2012.
- [5] Google Inc. Google code. <http://code.google.com/intl/da-DK/>, 2012. Last visit: 11-05-2012.
- [6] DynaVox Mayer-Johnson. Boardmaker software. <http://www.mayer-johnson.com/boardmaker-software/>, 2011. Last visit: 03-05-2012.
- [7] Yuku Sugianto. Android color picker. <http://code.google.com/p/android-color-picker/>, 2011. Last visit: 23-05-2012.
- [8] yuri.kanivets. The wheel widget for android. <http://code.google.com/p/android-wheel/>, 2011. Last visit: 23-05-2012.