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 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

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*[5]. To use the pictograms, the educators have to choose and edite 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. There have not been developed any similar android applications which target autistic children and their guardians. It is therefore 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*[4].

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

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 parts 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.
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 constrains for the applications, such as time constrains, 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 Paper Prototyping

Paper prototypes has been a tool 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.

Der skal sikkert lidt kilder og forklaringer til, samt reference til vores papirprototyper i appendix

3.3.1 Metaphors

To enhance usability and learn-ability, we have used metaphors 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 paperclip, which is known from the attach function in other programs, for example Microsoft Outlook Express.

Screenshot a vores metafor, og eventuelt en kilde til, hvorfor det er smart at bruge metaforer

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 section **??** all 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 functionality 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

Sprint 3: 10/04 - 19/04

Sprint 4: 23/04 - 04/05

Sprint 5: 07/05 - 11/05

Sprint 6: 14/05 - 18/05

CHAPTER 5

Implementation

Nice...

5.1 Fragments

In short is a fragment in Android 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:

This and more information about fragments can be found at [3].

5.1.1 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.

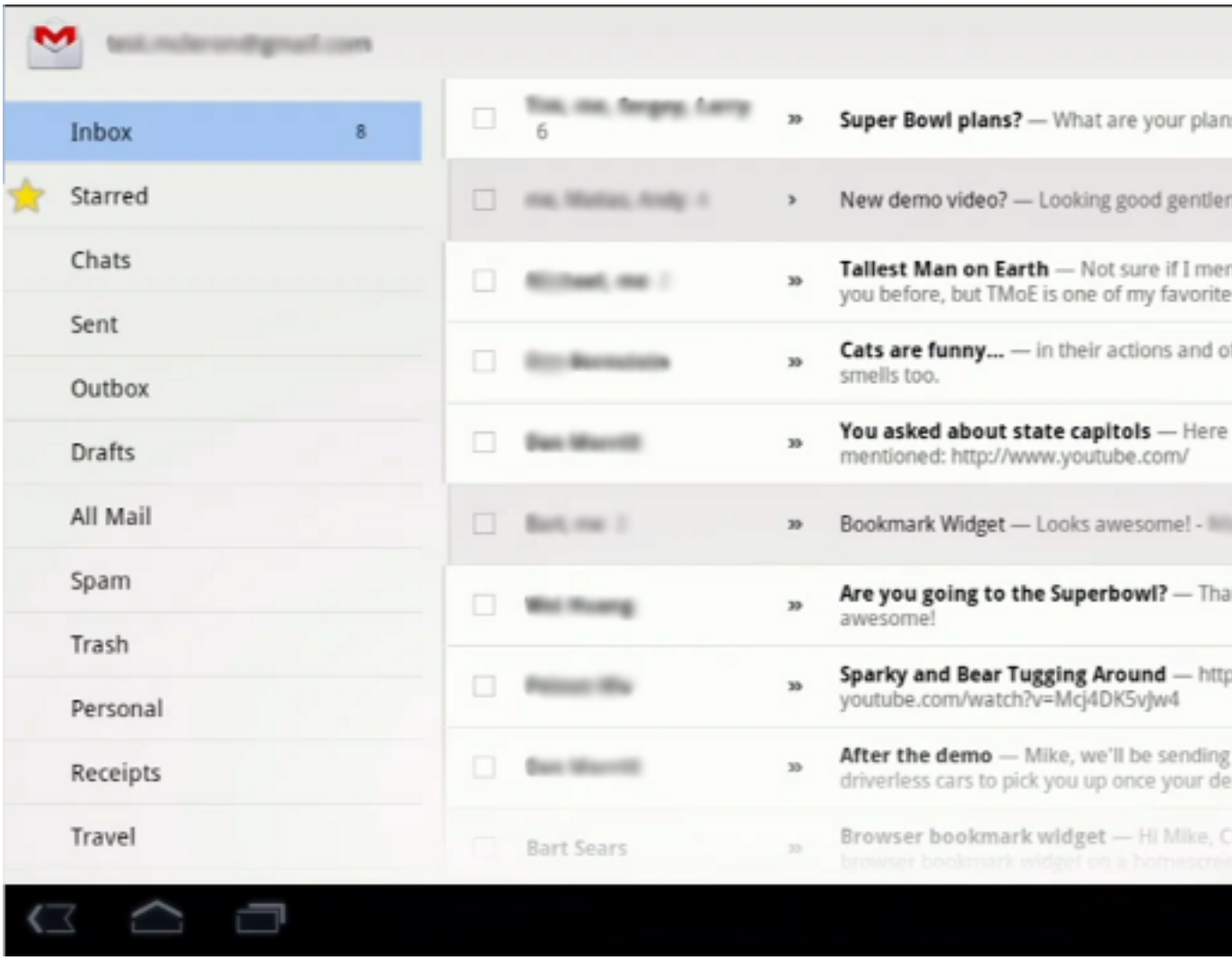

Figure 5.1: The Gmail application from Google on a tablet with fragment support.

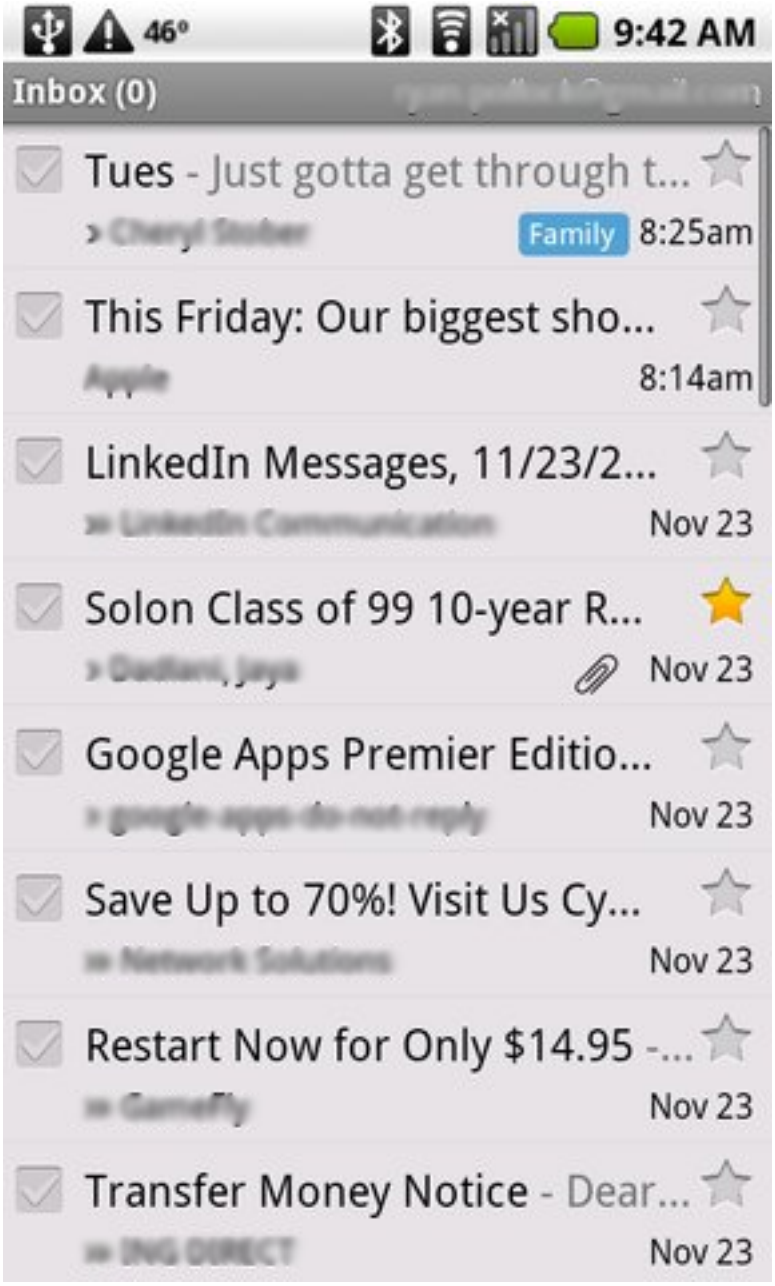

Figure 5.2: The Gmail application from Google on a handset without fragments

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.

As of 3rd March 2011 Google has made the "'Android Compatibility Package"', a library which let Android 1.6 devices or newer support fragments aswell [1]. Mobile tuts+, a website with tutorials about Android development, has tested the "'Android Compatibility Package"' and confirms that it even works on the T-Mobile G1 (the first Android device) [2].

5.1.2 Fragments in WOMBAT

5.2 Lists

5.3 Customize

5.4 Timers

5.5 TimerLib and DrawLib - Backend Library

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, and these are then out sourced to one of the other project groups of this semester.

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", 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 *stillHAfterSave#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*

CHAPTER 7

Test Results

Pass/fail table

Tail

Part III

Discussion

Head

CHAPTER 8

Reflexions and Evaluations

Usability test evaluation, evaluation of pair programming and refactoring, other SCRUM things

8.1 Conclusion

8.2 Future Work

Authenticate, port to cellphone, 3D drawings, Remake of design with drawer...

Tail

Part IV

Appendix

Appendix

Bibliography

[1] Tim Bray. Fragments for all. http://android-developers.blogspot.com/2011/03/fragments-for-all.html, 2011. Last visit: 15-05-2012.

[2] 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.

[3] Google. Fragments | android developers. http://developer.android.com/guide/topics/fundamentals/fragments.html, 2012. Last visit: 15-05-2012.

[4] Google Inc. Google code. http://code.google.com/intl/da-DK/, 2012. Last visit: 11-05-2012.

[5] DynaVox Mayer-Johnson. Boardmaker software. http://www.mayer-johnson.com/boardmaker-software/, 2011. Last visit: 03-05-2012.