

Valens - An Android application system to detect fall risk among the elderly

Vassdal, Johannes Willumsen	Aamot, Elias
johannes.vassdal@gmail.com	eliasaa@stud.ntnu.no

Larsen Tomren, Filip Andre	Pham, Dat-Danny
filip@tomren.it	datdanny@hotmail.com

Kücükyareli, Tayfun
tayfun.kucukyareli@live.de

April 19, 2013

Contents

1	Introduction	4
1.1	Group introduction	4
1.2	Customer introduction	5
2	Requirements	6
2.1	Final requirements	6
2.1.1	Functional requirements	6
2.1.2	Nonfunctional requirements	7
2.2	Understanding the requirements	7
2.3	Initial requirements	8
2.4	License	8
3	Testing	10
3.1	Levels, methods and types	10
3.1.1	Levels	10
3.1.2	Methods	11
3.1.3	Types	12
3.2	Plans and process	12
3.2.1	Unit testing	13
3.2.2	Integration testing	13
3.2.3	System testing (preliminary)	14
3.2.4	Acceptance testing	14
3.3	Tests used under development	14
3.3.1	GUI Tests	15

4	Project Management	17
4.1	Terms	17
4.2	Management Tools	18
4.3	Work Breakdown Structure	18
4.4	Development process	18
4.4.1	How Tasks are assigned	20
4.5	Software development	20
4.5.1	Agile software development	20
4.5.2	Iterative approach	21
4.5.3	Extreme programming	21
4.6	Group development method	21
4.7	Meetings with Customer and Supervisor	22
4.8	Plans	23
4.9	Team Roles and Organization	23
4.9.1	The group	23
4.9.2	Supervisor	23
4.10	Comparison	24
4.10.1	Conflict handling	24
5	Development Environment	25
5.1	LaTeX	25
5.2	Github	25
5.3	Eclipse with Android Development Tools Plugin	26
6	Implementation	27
6.1	Overall Architecture	27
6.2	Architecture of Valens Health Helper	28
6.3	Architecture of the Content Provider	33
6.4	Class description and diagram	33
6.5	Architecture of the Valens Step Detector	34
6.6	Step Detection Algorithm	36
6.6.1	Development Process	36
6.6.2	Core Algorithm	36

6.6.3	Application of the algorithm to live data	38
6.6.4	Constant values	39
7	Research and Alternate Solutions	41
7.1	Falling: Causes, Consequences and how to avoid it	42
7.1.1	Risk factors	42
7.1.2	Advice for prevention	43
7.1.3	Physiological aspects	45
7.2	Related Applications	46
7.2.1	Endomondo Sports Tracker	46
7.2.2	GPS Status	47
7.2.3	Pedometer	47
7.2.4	GPS Tracker	47
7.3	Summary of interview with medical professionals	48
7.3.1	Informal usability test	50
7.4	Lessons learned	50
A	Appendix	53
A.1	Previous WBS	53
A.2	Reports	53
A.3	Apache license and notice	57
A.4	Meeting summaries	63

Chapter 1

Introduction

Damage from falling is the main cause for hospitalization among the elderly. In addition to the enormous costs incurred to society, falling is also the cause of great personal tragedy. As the use of computational devices becomes more common, it seems natural to examine how such devices can contribute to preventing such accidents.

This project is a cooperation between SINTEF and a group of students at NTNU for the course IT2901. Its goal is to develop a model for risk level of movement based on common sensors found in Android smart phones, and build an API to make the model accessible for third-party developers.

1.1 Group introduction

The group consists of 5 members, four of which were from Norwegian University of Science and Technology, and the remaining student was an exchange student from Germany, Technical University of Dortmund. The members had experience with programming and project development with the programming languages Java and C, but little experience with Android applications.

1.2 Customer introduction

The customer was ”Selskapet for INdustriell og TEknisk Forskning ved norges tekniske hoegskole”, or SINTEF for short. The person representing SINTEF was Babak A. Farshchian, Adjunct Associate Professor at NTNU.

Chapter 2

Requirements

The requirements for the different parts of the project were about the requirements for the application, the underlying model and content provider, and the documentation for the model and content provider.

2.1 Final requirements

Several meetings with the customer and health experts led the initial requirements to be out of date and a little vague which led to new and updated requirements.

2.1.1 Functional requirements

- An Android content provider that stores and makes the data available for other applications through an API. The content provider shall have the ability to show requested data appropriately without any loss of information and in the way how the user wishes
 - Measure gait speed: The application shall be able to detect the speed of the user's gait so that this information can be used as an element in the whole detection process.
 - Measure gait variability: The application shall be able to detect the variability of the user's gait. That will be one of the main parts

for understanding its fallbehaviour and the reasons of his falling down.

- Measure steps: The ability to counting the steps in order to use their amount as a parameter for the walk/fall statistic.
- Filter out noise in terms of bad and unnatural movements.
- An example application that can visualize this data and represent them in a pedagogical manner.
 - Ability to do self-tests
 - * Reaction speed
 - * Stand up sit down capability
 - * Balance tests
- Documentation on the content provider that can be referred to

2.1.2 Nonfunctional requirements

- 80% of the elderly should manage to use the app within 10 minutes of education given that they have some prior knowledge of smartphones.
- Midly visually impaired should be able to use the application.
- Color blind should be able to use the application.

2.2 Understanding the requirements

The requirements were specified after several meetings with the customer. The understanding the group had of the requirements before meeting the customer was not sufficient to create a set of requirements. The requirements were also subject to change as the project progressed.

The meetings served their purpose, and filled in a lot of holes that were missing from the preliminary description:

- The development process in question ¹ uses a iterative approach. This essentially means that the requirements expands as the time goes, and to define the requirements from the beginning is impossible. This decision was made in collaboration with the customer.
- The team at SINTEF got a wide range of experts to help the group with health-specific features, e.g. making the algorithms to recognize movements.
- The customer expected weekly meetings with the whole group present, and that goals had been achieved before each meeting. At each meeting a new goal was set.

2.3 Initial requirements

According to the project description, the following should be developed:

- A model of physical movements based on common movement sensors found in Android smart phones.
- An Android content provider that stores and makes available the data in this model through an Application Programming Interface(API).
- An example application that can visualize this data.

2.4 License

The results of the project was intended to be released under the Apache 2.0 license. It is an open source license, which can be found at the following web site: <http://www.apache.org/licenses/LICENSE-2.0.html>

Highlights of what can, cannot and must be done under the license is described here:

Must be done

¹See Chapter 4.4 for a description of the development process

- include copyright
- include the license
- state what changes has been made

What can be done

- use copies of the licensed work without paying
- use the content with commercial products later
- modify the licensed work as desired
- the work can be distributed as desired

Cannot do

- trademark the licensed work

The specific terms can be found at the mentioned address, and has also been included in the appendix, see A.3, along with License and Notice files.

Chapter 3

Testing

This chapter only touch the surface of general software testing, while focusing on methods, types and levels used in this project.

3.1 Levels, methods and types

Software testing is broken down into several levels, methods and types.

3.1.1 Levels

Software testing has four levels that corresponds to how far into the software development process the team has come. The first three test levels refers to the development process and the last level refers to completion, or after development.

1. Unit testing is the first level where all individual components, such as functions, are tested. Often these tests are done by inputting sample data and validating output.
2. Integration testing is the next steps which takes groups of individual components to verify that they work together or to reveal faults in the integration.
3. System testing is the last step of test in the development process where

the complete system or software is tested to evaluate it's match to it's requirements.

4. Acceptance testing is the last and highest level of testing where the system or software is tested to verify that it is acceptable for delivery. The test itself is usually Black Box Tests (see page 11) to see if the customer will receive the desired and expected results.

3.1.2 Methods

The approach to software testing are called methods or techniques. Primarily there are two methods when testing software; black box testing, and white box testing.

- Black Box Testing is named so because the internals of the software or system is not visible to the tester, like inside a dark black box. The tests are usually functional tests to reveal interface problems, functionality flaws, performance troubles, behaviour errors and incorrect or missing functions. This test method is usually done by independent testers with no programming or implementation knowledge. Mainly these test method is applicable at a higher level of testing; System testing and Acceptance testing.
- White Box Testing is a test method where the internals of a software or system is known to the tester, who usually is a programmer. Programming and implementation knowledge is required in this method since this method studies the code and goes way beyond the interface visible to the end user. The method is mainly applicable in lower levels of testing like Unit testing and Integration testing, but it is mostly used in Unit testing. The overall advantage of this test method is that testing can be started on in an earlier stage to make sure every bit of program produces correct output given an input and gives the opportunity to quickly correct problems.

3.1.3 Types

Only some of the test types are mentioned in this chapter, though there are many different testing types not used in this process and therefore not covered.

- Smoke Testing, only tests major functions of a software or system to decide if it further testing should proceed. If the Smoke Tests fail, no further testing is necessary due to deeper testing will fail. A kind of Smoke Test could be to check if the software compiles and starts up. Smoke Tests should under no circumstance replace Functional or Regression Testing.
- Functional Testing, verifies if the software or system to have met its requirements or specifications. In functional testing, Black Box Testing is performed where the tester has no aspect of what is going on internally. See above explanation about Black Box Testing for more. When performing Functional Testing a set of inputs and expected outputs are matched up to one another, where they do not match the test fails. When performing this type of testing it is easy to miss logical errors due to the fact that the tester do not know what is going on underneath the code.
- Regression Testing, essentially re-tests previously verified tests to ensure that bug fixes, enhancements or optimization has not affected the system or software after a given change in code. The test type can be performed on any level, but is mostly relevant on the System Testing level when the bug fixing, optimization and enhancements are done. Many companies choose to automate these tests to save time and money when software or systems has changed.

3.2 Plans and process

During the first period of development with the incremental process, test plans were not laid out due to the details of the process with no known end-

results which would have added a lot of extra work redesigning testing plans after every customer meeting. After switching to agile development process testing got a lot more focus as described below.

3.2.1 Unit testing

Due to the, at first, incremental development process, unit testing was done on-the-fly to make sure that every methods worked as they should. When a code change in a specific method was done, the unit was again tested to verify new outputs. To make a test plan when developing incremental seemed unnecessary since the requirements changed and methods was created and removed all the time.

When the process was changed to the agile development process, unit testing became more important to verify the output of each method during development, though the documentation and test were added after finishing a method. A unit test was never sufficiently documented before the initialization of method development. Please refer to Section 3.3.

3.2.2 Integration testing

The integration of this project is divided into two levels; method- and system integration. Due to the fact that this project essentially consist of three standalone application every application was tested for themselves and then finally together.

The first integration testing was every application for themselves. This was done by just trying to run the application and checking the outcome after every run and was mostly done by the developer themselves. The reason to choose this kind of approach to this form of testing was the fact that an integration test plan takes time to write and every application was very compact with few functions. There are of course negative sides of this kind of approach that includes lack of useful documentation, overlooking non-obvious failures and not knowing exactly how the final application should work. The alternative was to have a test plan worked out beforehand to follow and

to explain all of this, but due to the lack of knowledge about testing and incremental processes this was neglected due to the reasons stated above.

The second part of integration testing was the system as a whole while developing. This was also done by just trying to run the applications and checking that the data flow between the applications worked. In this stage a form of Smoke Testing were used by checking if the data storage increased (data being saved to the phone) and by checking the output in the GUI app (statistics screen showing data).

3.2.3 System testing (preliminary)

The second to last testing level were done almost parallel to the second part of the integration testing to verify that everything was working together and that it met requirements. Differing to the previous level of testing this part had tests laid out beforehand.

The System Testing is although done a little bit after the main developing, where it was just testing all parts of visible software (Black Box Testing). All the faults, bugs and defects noted here were noted according to the tests laid out and fixed by a developer before further testing.

3.2.4 Acceptance testing

The last level was before handing over the final application to the customer, but testing in a production environment. The same tests were performed as in the System Testing level and bugs, faults and defects were again noted to be fixed before rolling out a new version for testing. After no tests failed the application were considered stable and delivered to the customer.

3.3 Tests used under development

Here goes the tests for the GUI and demo application:

3.3.1 GUI Tests

Here is the tests for the demo application and the GUI.

Info					
Title	Activity start				
TestID	01				
Writer	Johannes				
Developer	Dunno				
Date created	date test was planned				
Date tested	date test was performed				
Tester	Everyone				
Preceding test	NA				
Test data	Application and android phone or emulator				
	Description	Input data	Expected result	Real result	Status
Step 1	Start up application in phone or emulator	Nothing, just program start	Write name screen	Write name screen shows up	Correct!
Step 2	Write name in name box and click OK	A valid name	Greeting with name	Greeting with name	Correct!

Info					
Title	Event list and details				
TestID	02				
Writer	Johannes				
Developer	Dunno				
Date created	dunno				
Date tested	dunno				
Tester	Everyone				
Preceding test	Test 01				
Test data	Notifications stored in database				
	Description	Input data	Expected result	Real result	Status
Step 1	User tries to access notifications	User clicks smiley face	List of notifications is displayed	List of notifications is displayed.	Correct!?
Step 2	User clicks on notification title	User click and database input	Detailed view of notification	Detailed view of notification	Correct!

Chapter 4

Project Management

4.1 Terms

Here follows a description of terms that are useful to understand how the project management functioned.

Scrum is an popular agile software development methodology. This means that the work is done in increments called sprints, and daily meetings are held to update the rest of the team on progress and problems. The scrum process also has roles for team members, in particular a team member called scrum master who handles contact between the development team and people outside the team.

Sprint is a period in which work is planned and done. It has a duration of one week. At the end of each sprint the group is updated on progress achieved, and sets goals for the next sprint.

Kanban-board is a visual aid that was used to display work packages, with content, people assigned to each package, and status of work package (to do, currently worked on, finished). The intent was to simplify and organize the work being done and work in need of finishing.

4.2 Management Tools

Trello was a collaboration tool with the ability to create interactive kanban-boards online. This use of this tool was to allow the group to coordinate tasks that were to be done, and the progress on the tasks, and which members were to work on which task.

It's learning was used to distribute information that was not time -critical, with a message board being used. It's learning would not send messages when a new topic or message appeared, so it's use for time-critical messages or making sure that everyone would read it was limited.

Email was used for time-critical communication, and for information that needed feedback swiftly, often within the same day.

Github was used to share code, and to describe and mark issues found in the code when problems were discovered. The relevant issues could then be discussed on the website

4.3 Work Breakdown Structure

With a development methodology that stresses short term goals, it is only possible to plan WBS for the current period. After the customer has specified the goal for the following week, we immediately sat down and defined the WBS for the given period. This was, a WBS was developed incrementally by adjoining the new WBS to the previous. The WBS was made in graphical form, as can be seen in Figure 4.1.

4.4 Development process

The customer favored the use of an iterative approach to the development process, where every sprint added a new layer of functionality, either to the application or the underlying model. Each sprint lasted for a period of 1-2 weeks, and the exact content was worked out in collaboration with

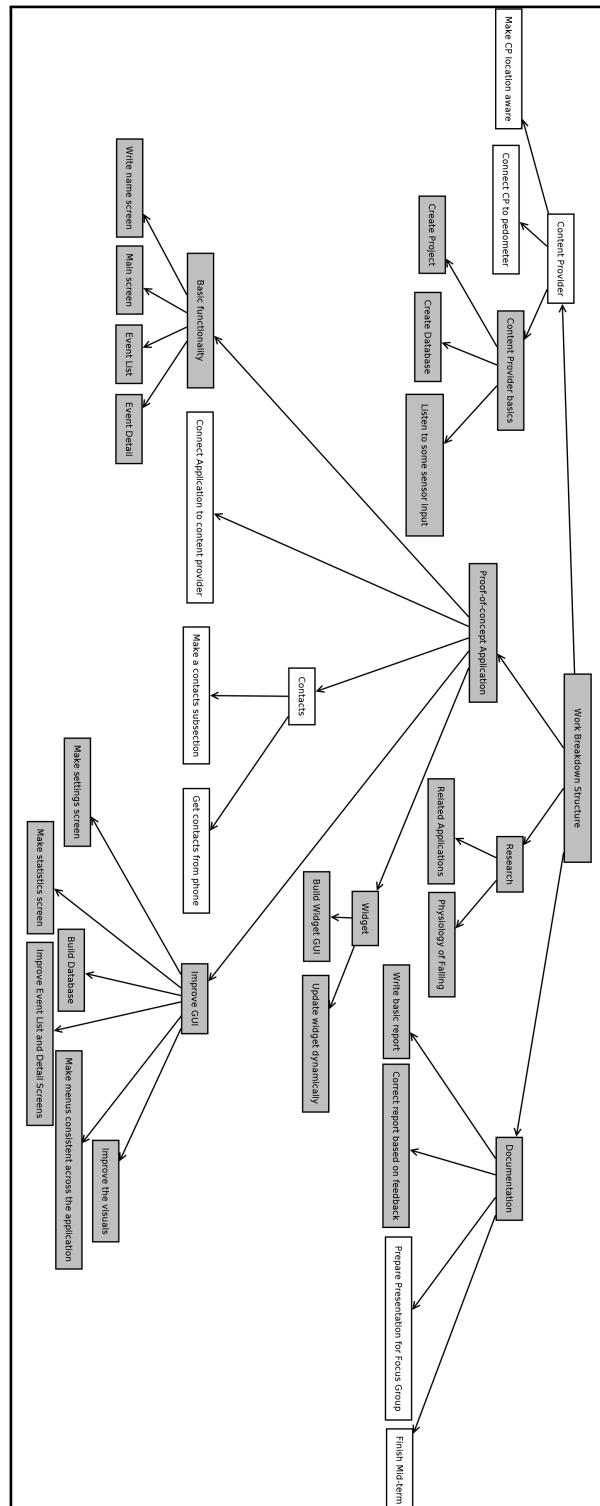


Figure 4.1: The current WBS

the customer. Short term plans were favored over longer plans, due to the flexibility provided. While this made formulating definite goals for the final product difficult, the customer and the group were in agreement that due to the research intensive nature of the project, a high degree of flexibility was required.

It was decided that the developers would have online meetings twice a week and an offline meeting once a week. The working hours were set to not less than 20 hours a week, but the developers were free to choose when to work themselves. This number was the expected workload for the course, and made a reasonable target for work per week. The meetings with the team was set to two times per week instead of once per day, so it could fit with the schedules of the team members. This was different from other agile methods like Scrum, but it was a necessary adaptation to fit meetings into the entire teams schedule. Updates were shared among the team members with email, Trello¹, and it's learning as a replacement for daily meetings.

4.4.1 How Tasks are assigned

As a working platform the group is using the website trello.com where the tasks are updated every week. Here tasks can be declared in a table of labeled with tasks "to do in future", "doing" and "done". This is done by Elias, although all group members could ask to work on one part in particular. When the work is done, it is reported back to the board on trello, and a new task can be assigned.

4.5 Software development

4.5.1 Agile software development

Agile software development is a collection of software development methods. These methods were developed based on iterative and incremental software development. Agile development also cuts the working process into short

¹See page 18 for more info

periods of working time, which gives us the possibility to see the progress and act accordingly. Agile development focuses on evolutionary development, where the team can make a plan according to what has been done.

4.5.2 Iterative approach

Iterative development simplified is a cycle of agreement, execution and assessment. Most engineering project using this approach also includes stages like requirements and design. The project using iterative development will enter several stages where these stages is repeated in a cycle formation. The cycle will usually contain a planning phase, which makes it relatively easy to react to a current problem, this can for instance be time spent and time left. The software is developed through several rounds with these cycles.

4.5.3 Extreme programming

Extreme programming is an agile software development methodology which favors improving of a software and quick responses. Extreme programming uses a cycle based programming system, but unlike the Iterative approach, it has a checkpoint after each stage, where the code is being released and new customer requirements is added.

4.6 Group development method

At the beginning of the semester the group were given a task to develop an application system. At that time they did not have any idea on how the system as a whole would look like, but they were given small goals every sprint like developing specific parts of the application system. At this time they used a development method almost equal to Extreme programming, but took some concept from Scrum since they were more familiar with terms from Scrum like sprint. The method used gave possibilities for quick responses like editing the code to what the customer wanted or changing algorithms to more effective ones.

After Easter the group got a clearer vision of their final goal, and the development method were switched to Scrum development method. Clearer sprints were formed. Smaller task with estimated time usage were formed on Trello, as Trello were used more frequently than before. Tasks with similar working area were put together in a bigger assignment, but with the possibility to only do the one small task if they wanted to. After making the tasks the Group leader and the document-organizer could freely assign tasks to the rest of the group members, and tasks that had not been assigned were for those who would finish early with their work. This was done to organize the tasks to get a better overview for better planning.

4.7 Meetings with Customer and Supervisor

The main essence of the project was that it is based on an incremental working model. Because of this, the project needs regular updates by the stakeholder instances(customer,supervisor) for requirements and functionality. For this reason meetings between the group and the customer took place every Friday with the customer and every second week with supervisor. From these principles the project benefits in many aspects:

- The project stakeholders is kept up-to-date
- The progress of the project is predictable
- The project gets feedback before and after the tasks are done
- The project is split in small pieces of actionable and specific smaller tasks
- These smaller elements can be discussed and considered before they are started
- Due to the smaller pieces of progress, mistakes can be backtracked with a minimum of wasted work
- Customer gets to participate in the development

- The group can take action when the supervisor or customer gives feedback on the results

4.8 Plans

Nearing the end of every sprint, the customer and group agreed upon the content of the following sprint. This list has been placed in Figure A.3, and provided a short summary of what was accomplished in a particular time-frame.

4.9 Team Roles and Organization

4.9.1 The group

There was not much place for specific roles among the group, as it was a small group, and the project required that all the members were capable and willing to work at all the tasks. This makes a difference from development models with specific roles and tasks assigned. Roles that were set for the group were therefore mainly organizers, so that one person was to keep awareness of what work needed to be completed in a particular domain, and share it with the rest of the group:

Group Leader Elias was made organizer, and got the responsibilities of reminding the group of what to do.

Document-organizer Johannes was tasked with organizing documentation and distributing the work of writing the report to the group.

4.9.2 Supervisor

The developers were assigned a supervisor from the institute. The supervisor, Tinna Tómasdóttir, gave guidelines and feedback on how to build the reports and make documentation.

4.10 Comparison

The development model used by the group was a form of agile development. Other well-known examples of agile methodologies are Scrum and Extreme Programming.

Chapter 5

Development Environment

5.1 LaTeX

LaTeX is used to write this report. LaTeX is a typesetting language, with support for varied formatting, including images and including other document. It also has the option allowed for inclusion of multiple LaTeX documents (.tex documents). This leads to improved readability of LaTeX code and productivity, as several chapters could be altered at the same time. This functionality allows a group of multiple users to avoid the problems associated with multiple users writing in the same document.

5.2 Github

It is requested by the customer that the group use the tool Github to share code and perform version control. Github has browser-based interfaces, download-able clients, and a robust command-line interface, meaning all the members of the group can make use of it. There are some problems associated with learning how to use and fix problems with it, as not all group members have experience with these tools.

5.3 Eclipse with Android Development Tools Plugin

The demands from the Integrated Development Environment(IDE) are as following:

- Has support for programming applications for Android
- Is understood by at least some members of the team

To fulfill these requirements, and because there are a plenty of tutorials that can be found, the group decides to use the Eclipse IDE, with add-on's for easier code and deploy towards Android.

Chapter 6

Implementation

6.1 Overall Architecture

The product has been implemented as a system of independent component applications where each component performs a well-defined task. The motivation behind the modular architecture is two-fold: Primarily, it was desirable that the underlying model for movement and fall risk be independent of the proof-of-concept application, in order to facilitate development of possible future applications employing the model. Likewise, it is desirable to enforce independence between the content provider and the sensor model. Secondly, the standards of Android application development state that a Content Provider should in itself solely provide an interface to the data. This implies that the sensor model, as well as any derivation of secondary data, should be performed in components independent of the Content Provider.

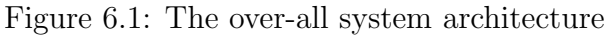
The system that has been designed according to these principles therefore consists of five components: The content provider *Valens Content Provider*, the proof-of-concept application *Valens Health Helper*, the sensor model *Valens Step Detector* and a service of deriving secondary data *Valens Content Feeder*. Each of these components fulfils a well-defined role in the total application system: The content provider provides an interface to the data model. The sensor model listens to the sensor data, and feeds the content provider with the timestamps of any detected steps. The Content Feeder listens continu-

ally to changes to the data in the content provider. When the data changes, it uses the new data to calculate secondary data, such as gait speed and variability. Finally, the proof-of-concept application demonstrates one simple way in which the data provided by the content provider can be used as a part of a health-promoting app.

6.2 Architecture of Valens Health Helper

The application is made in a way that is common for all android applications. This means that user interface is described in XML layout files that is called in Java code. Strings and resources is placed in a separate folder and file, to be accessed by the code as needed. This is to separate content and layout in the UI. The separation of layout and strings enables different localizations, so that the application can provide a user interface in different languages easily.

As is common in android applications, classes that inherit from the class Activity define a separate screen in the GUI. Because of this, a significant part of all the classes in the application are activity classes. These classes simply define the behaviour of their GUI screen. The flow for the GUI can be seen in Figure 6.2, and an early mockup of the GUI can be seen in Figure 6.3.



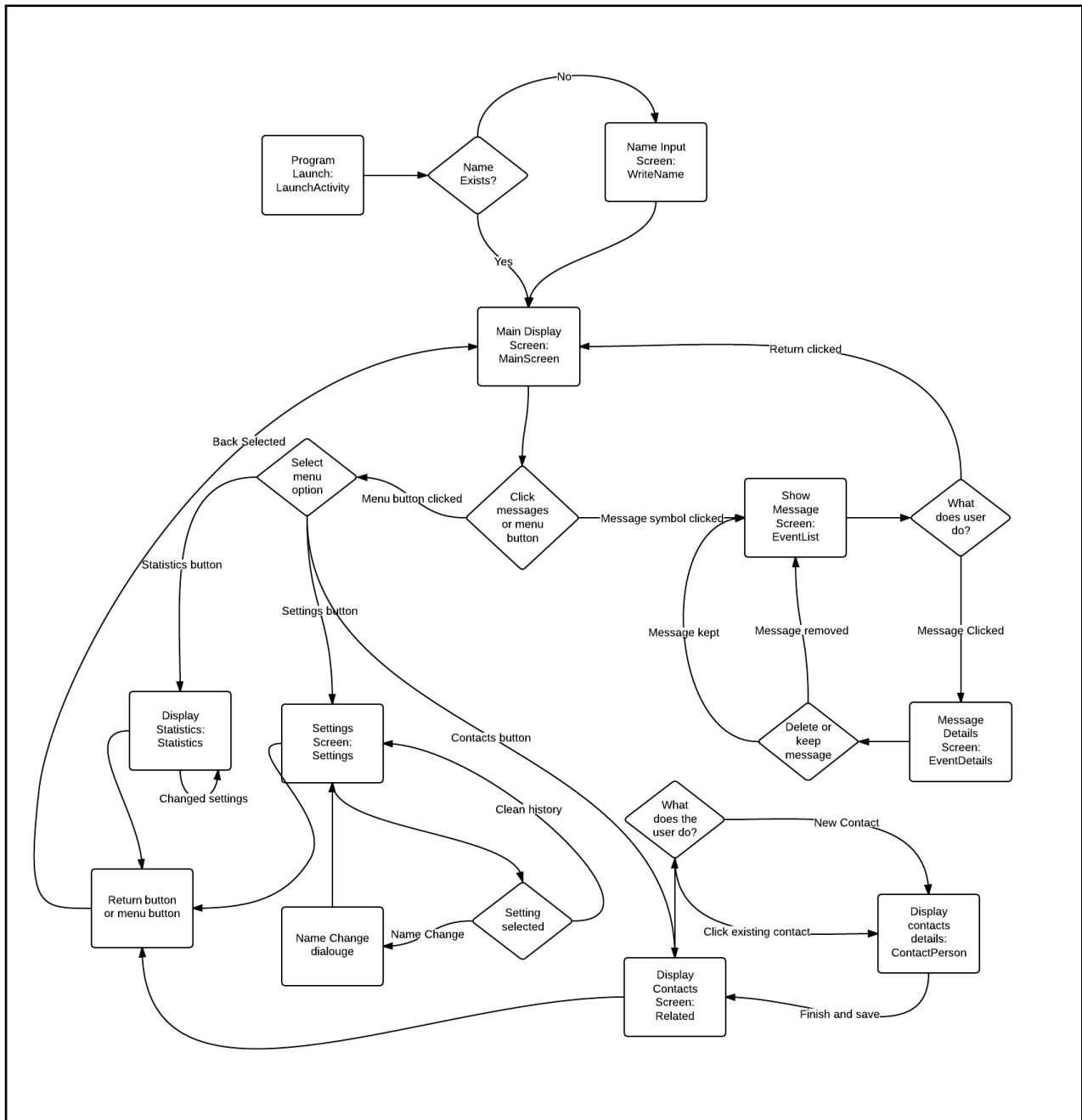


Figure 6.2: Flow chart describing the Graphical User Interface

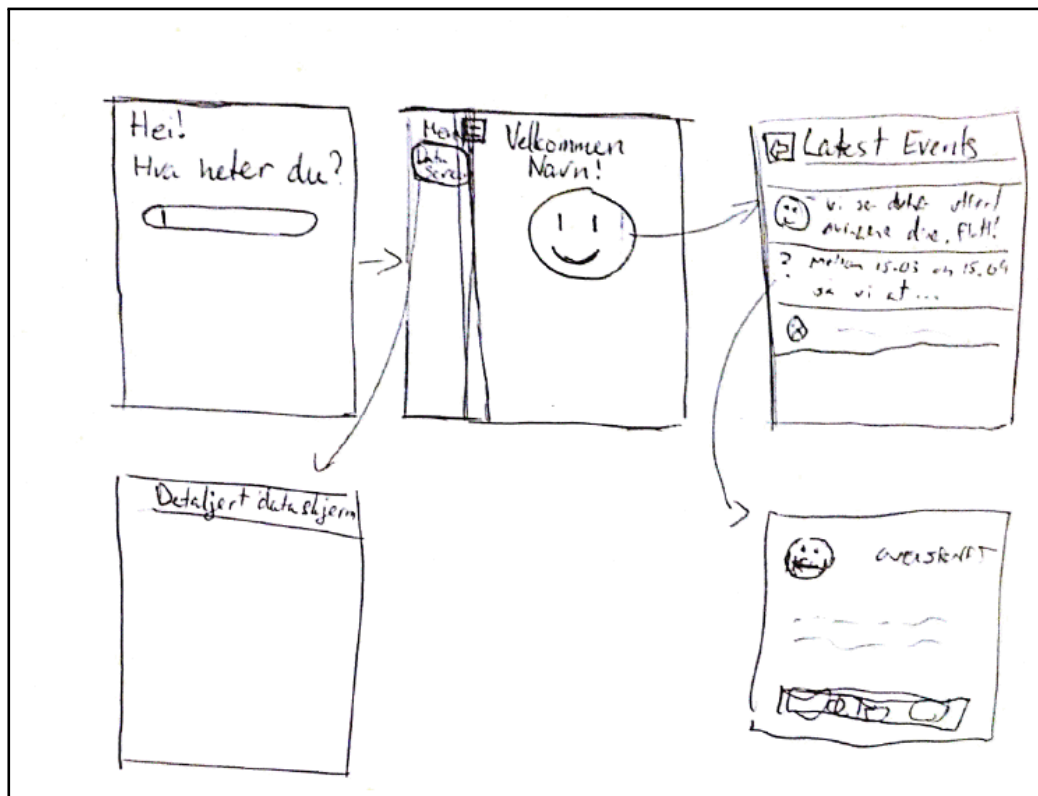


Figure 6.3: A mockup of the program flow, text is just scribbling

Classes that are not activities can be roughly divided into 5 types:

Activity classes define the activities employed in running the app. In Android, an *Activity* is "a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI"¹. Every GUI screen corresponds to an activity, but not all activities need to have a GUI. In the current implementation, every activity except for *LaunchActivity* has a GUI screen, so see the GUI flowchart for an overview of the activities used in the current implementation.

Connectivity classes provide an interface for communication with the database and content providers. There are three connectivity classes:

¹<http://developer.android.com/reference/android/app/Activity.html>

ContentProviderHelper handles communication with the content provider. In particular, it contains several methods that sends queries to the Content Provider, interprets the data, and returns useful values to the GUI activities.

DatabaseHelper handles communication with the database. The database contains information necessary for the demo application to function, in particular events, messages and contact information.

DatabaseContract defines the strings used for table and column names in the database. While a DatabaseContract might seem redundant, it is common android application practice to use one. A database contract functions as a "contract" the define the legal interaction between the database and the application, by stating the legal table and column names.

Data structure classes each define a useful data structure in the program. The data structure classes in the app are Contact, Event and RiskStatus.

List adapter classes define how elements in a list should be displayed, and what kind of data should be associated with each element.

Widget classes define the functionality of the widget. There are two widget classes:

WidgetProvider creates the widget and provides it with a GUI. Is called by the Android OS when the user creates the widget.

WidgetUpdateService takes care of updating the widget a regular intervals, by looking for new information in the content provider. Handles regular check with the help of an internal TimerTask class, Updater.

6.3 Architecture of the Content Provider

In android, Content Providers provide an interface to structured data of some sort. Access to for instance the list of contacts in the phone, or the phone's calendar is managed through standard content providers. Content Providers provide methods for inserting, updating and deleting relevant data. A major part of the assignment was to implement a Content Provider that gives developers access to structured movement data.

The Content Provider component is conceptually very simple, and consists of four classes:

CPValensDB a subclass of SQLiteOpenHelper, the standard android class for handling database communication. Specifies the procedure for creating the database, as well as upgrading and downgrading the database version. In the current implementation, upgrading and downgrading the database version clears all the data and tables, and creates the database according the details specified in the new database version. In the current version, creating the database consists of creating the tables given in the data model, without feeding any data into the database.

DBSchema defines the database model and provides string values for the names of the tables and fields. Employing a database schema to demonstrate the structure of the database is standard in android applications.

ValensDataProvider a subclass of ContentProvider, which is the standard android class for implementing content providers. This is the core of the content provider, and describes how queries to the content provider are handled.

Main provides the content provider with a basic GUI.

6.4 Class description and diagram

The class structure of the Applications changed regularly, and the most recent description is in Figure 6.4. Older diagrams and descriptions was placed in

the appendix.

6.5 Architecture of the Valens Step Detector

Despite being a relatively simple application in terms of lines of code and GUI screens, the architecture of Valens Step Detector has a certain complexity.

LaunchActivity handles the basic GUI for the main screen.

Values holds the values of the constants used for step detection.

Methods contains most of the functions used for calculation of peaks.

StepMainService is main class for the sensor model. Is implemented as an Android Service, and as such, it runs in the continually in the background until the user explicitly stops it. Functions as a SensorListener that listens to input from the accelerometer, and keeps track of the sensor vector lengths along with their timestamp. When enough data has been generated² the StepMainService will launch a DetectStepsThread to detect steps in the given data, before clearing its data³ to ensure that the maximum memory required by the service remains constant.

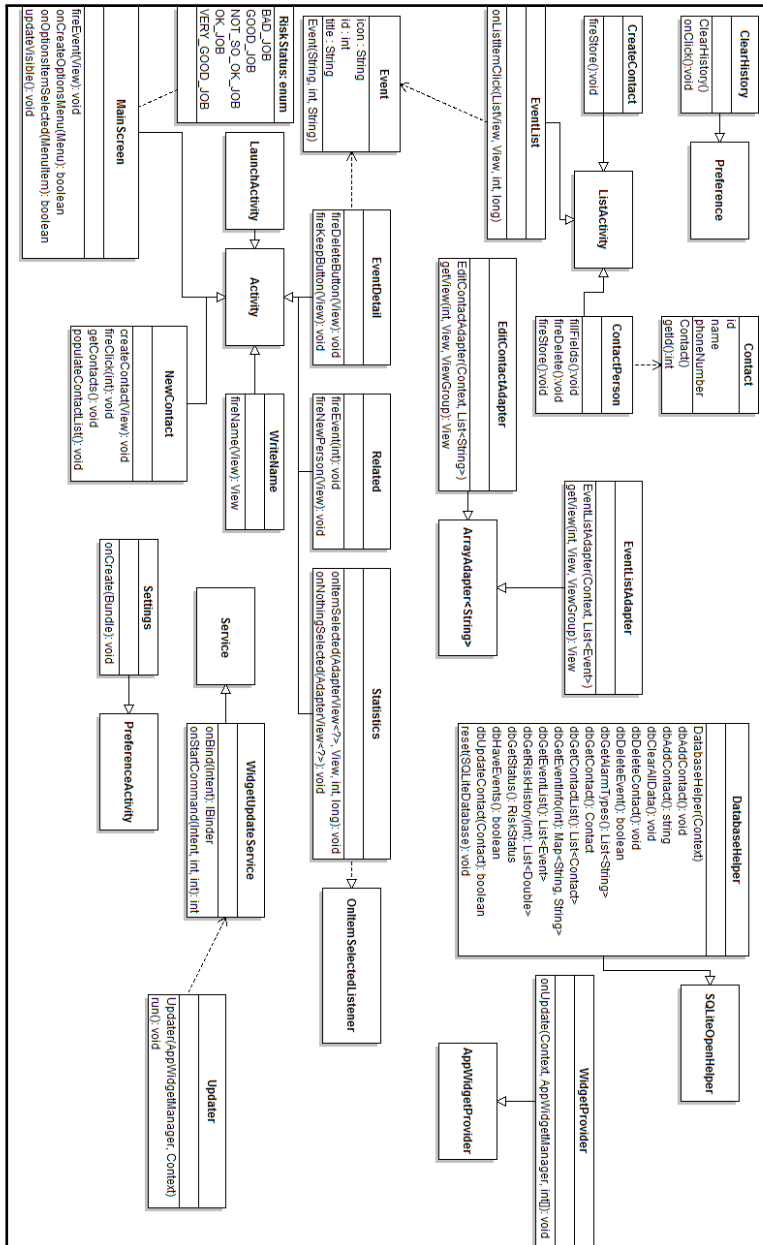
DetectStepsThread searches asynchronously for steps in the data provided to it. Step detection is performed in a separate thread so that sensor input is not interrupted by the calculation.

Calibration Classes are a group of classes that are used for calibration, that is adjusting the mean and standard deviation to the step patterns of the user. There is one GUI class, one class used for timing and and one sensor class:

²Enough data is defined as $m + 2k + 2n$, where k is the size of the smoothing window, n is the size of the peak strength window, and m is a constant which defines how many time steps one wants to use for step detection. As the first and last $k + n$ steps are not used for step detection (see explanation of the algorithm), only m values will actually be used for step detection.

³For reasons that will be explained in the description of the algorithm, $2k + 2n$ data points are retained.

Figure 6.4: A class diagram of the current application



CalibrationActivity handles the GUI functions for the calibration screen. When the "calibrate"-button is pressed, it starts a timer that starts the **CalibrationStartTask** after a certain amount of time has passed, so that the user has time to lock the screen and put the phone in his/her pocket.

CalibrationStartTask is a **TimerTask** that simply plays a sound to indicate that calibration commences, and immediately starts an **CalibrationThread**. This class only exists because Java requires a **TimerTask** for the **Timer** to work properly, and **CalibrationActivity** can't inherit from **TimerTask** as it already inherits from **Activity** and Java does not support multiple inheritance.

CalibrationThread handles sensor input during calibration. When the calibration time has ended, it stops receiving input, calculates μ and σ , and terminates.

6.6 Step Detection Algorithm

6.6.1 Development Process

As the group had little experience in sensor use and signal processing, a series of experiments were conducted to better understand which sensors provide useful information for step detection, and which patterns predicate steps. To this end, a simple app was developed to observe sensor input and store the raw data as a file, which then could be transferred to a computer. A set of python scripts were written to plot the data time series. Using python, the group experimented with different step detection algorithms, and a prototype was developed. Finally, the algorithm was implemented in the Valens Step Detector app, modified to work with live data.

6.6.2 Core Algorithm

The current section will explain the algorithm used for step detection, formulated as an off-line algorithm. The following section will explain what

changes was made to make the algorithm work for live data streams.

Experiments showed that the accelerometer consistently gave the most predicative feedback patterns for step detection⁴. Furthermore, it was found that the total acceleration was a better predictor of steps than acceleration along any of the single axes individually. To exploit this, a pre-processing step calculates the vector length in euclidean space of the raw acceleration vector.

Detecting steps based on a time series of acceleration vector lengths reduces to peak detection in the time series graph. The step detection algorithm is therefore essentially a generic algorithm for peak detection in noisy data. The noise in the data originates from noise in the sensor readings. To cope with the noise, the data is first smoothed using a moving average⁵. A larger value of k gives a smoother curve, but a too large value of k may flatten the graph too much.

Subsequently the program runs the main peak detection algorithm, which consists of the following steps:

Calculate peak strength

Every data point in the time series has its *peak strength* calculated. Informally, the peak strength of a data point is a measurement of how much that point is worthy of being considered a peak. A wide range of functions can map the raw vector lengths into a peak strength graph, but an additional desideratum of the peak strength function is that the output values should be normalized around 0. With the peak strength function used in the current implementation the peak strength of the point x_i is calculated as

$$\frac{\frac{(x_i - x_{i-1} + x_i - x_{i-1} + \dots + x_i - x_{i-k})}{k} + \frac{(x_i - x_{i+1} + x_i - x_{i+1} + \dots + x_i - x_{i+k})}{k}}{2},$$

where k is a constant referred to as the *peak strength window*.

⁴None of the android phones available to the group at the time had a working gyroscope. It is suspected that a gyroscope can replace or supplement the accelerometer for more precise step detection, but this should be tested empirically.

⁵That is, every data point has its value set to the average of itself and the k neighbours before and after it, where k is a constant referred to as the *window size* of the smoothing.

mean and standard deviation]

The mean, μ , and standard deviation, σ , of all positive peak strength values are calculated. Sub-zero values are discarded.

Search for potential peaks

Every data point in the peak strength series is classified as either a potential peak or discarded. A data point i is classified as a potential peak if its peak strength value x_i is positive and fulfils the following inequality: $x_i - \mu > k * \sigma$, where k is a constant, referred to as the *std threshold*. The larger the value of k , the fewer data points are classified as peaks. This step thus functions as a high-pass filter, where the cut-off threshold is defined dynamically by μ and k .

Remove close peaks

If two data points that have been classified as potential peaks are very close to each other, it is likely that they both belong to the same peak in the actual data. To avoid peaks in the data from being classified multiple times, this step runs through the list of potential peaks, and for every pair of potential peaks it removes the least strong peak if they are closer than some constant k . In the current implementation, k is set to the same value as the peak strength window, following the approach taken in [?].

The data points that remain after the application of the algorithm are counted as steps.

6.6.3 Application of the algorithm to live data

The algorithm as stated above works on an off-line time series of data, and adapting it to work with live data requires some non-trivial decisions to be made, most of which will be explained and elaborated upon in this section.

While it is possible to change the algorithm to run completely on-line, classifying every new vector length data point as either a step or not, thus reporting steps exactly as they happen, this approach is problematic. Firstly, as no data exists past the newest data point, only the previous data points can

be used for smoothing and peak strength calculation. This can potentially reduce the precision of smoothing and peak strength calculation significantly. Therefore, a semi-live approach has been taken, in which a fixed number of data points are collected before step detection is performed. The collected data is then discarded, and the process start over again. However, some care needs to be taken to ensure that all the data is used, and no data points are used multiple times. The first and last k (the smoothing window) data points cannot be used for step detection, because there is not sufficient data to smooth them properly. Likewise, the first and last n (peak strength window) data points cannot have their peak strength calculated properly, and are not used for step detection. Because of this, the last $2n + 2k$ data points are retained when the collected data is discarded, as the last $n + k$ data points have only been used for smoothing and peak strength calculation, not for step detection, and the $n + k$ data points before that - which have already been used for step detection - are required for smoothing and peak strength calculation in the next batch of data.

Another issue when working with live data is how to calculate μ and σ . First of all, calculation of σ requires the entire data series, but retaining the entire series of raw data in main memory is infeasible. Also, in a real life application long periods of inactivity will reduce μ to unnaturally low levels, potentially creating false positives during step detection. A better solution is to base calculation μ and σ on actual movement data, so the values are adjusted to the relevant movement characteristics of the person. Based on these motivations, the app first requires the user to calibrate it by walking for 30 seconds. Values μ and σ are thus calculated once based on the time series generated during calibration.

6.6.4 Constant values

The constants used by the algorithm play a significant role, and finding good values for these constants is crucial for the performance of the algorithm. However, there it is impossible to know a priori which values give the desired result, but it should be noted that some of the constants (particularly the

window size constants) are related to the frequency at which sensor data is generated⁶. Because of this, most constant values have been found by empirical studies during the prototyping step, so there might exist room for improvement, but thorough empirical studies or machine learning techniques may be required to uncover better values for the constants.

⁶That is, the faster the sensor data is generated, the wider the windows can be without increasing the risk for mixing information between separate peaks.

Chapter 7

Research and Alternate Solutions

During the planning stage, the groups was able to identify two areas which required research:

Domain knowledge: No member of the group had any familiarity with the application domain, neither with the specific domain (fall studies) nor any of the related domains (like healthcare, medicine or sport science). A basic understanding of the domain is critical for communication with the expert group, as well as for the ability to develop and assess solutions independently of the expert group.

Related software: Related applications function as sources of inspiration, proofs of what is feasible to implement or even as aids during programming. Open source software can be partially or entirely incorporated into the code. Knowledge of related software can thus increase development efficiency.

For each of the two areas of research, one member of the group was assigned to research and compile a concise report, so that the rest of the group could efficiently attain the required knowledge.

The two reports will be reproduced in the following sections.

7.1 Falling: Causes, Consequences and how to avoid it

This report collected information from several sources on the subject of falls among the elderly. Particular focus is given to risks and avoidance strategies.

7.1.1 Risk factors

There are several factors that predicate risk of falling. Typical risk factors are:

- **Biological risk factors (sorted by order of significance in [1]):**
 - Muscle weakness
 - Balance deficit
 - Gait deficit
 - Visual deficit
 - Mobility limitation
 - Cognitive impairment
 - Impaired functional status
 - Postural hypertension
- **Behavioral risk factors(Not in sorted order):**
 - Inactivity [5]
 - Medications [5]
 - Alcohol use [5]
 - Living alone [6]
- **Home/environmental risk factors(not in sorted order): [7]**
 - Bad footwear or clothing.
 - Dangers in the house or in public places.

- Unfamiliarity with walking aids such as canes, crutches, walking chairs.

A fall is normally caused by the interaction of two or more of these risk factors, but home or environmental risk factors cause only 30-50% of all falls[5, 1]. What this means is that more than half of all falls happen without any influence of environmental factors. Also, in most of these cases where external factors play a role, the fall is in reality caused by an interaction between these and physiological aspects. The second and third most common single causes of falling are gait/balance disorders and dizziness/vertigo, followed by *drop attacks*, which are sudden falls without loss of consciousness or dizziness[1]. As the project task is to develop a model for physical movement, it might be necessary to down-prioritize identifying risk for dizziness/vertigo and drop attacks, as well as ignore and external factors, in order to focus on purely physiological aspects, such as gait or balance.

Many older adults are unaware of their risk factors, and therefore unable to take preventive actions. Even older adults with a history of falling have normally been given little education about the potential risk factors. Any sort of risk assessment, is therefore be very beneficial, especially when the results are discussed with a healthcare provider[5]. This fact illustrates the usefulness of planned app.

It has been showed that many of the biological risk factors can be reduced by effectively. Specifically strength, gait and balance has been shown to be improvable by simple exercise regimes - even for frail patients - in a number of studies[5, 7, 2]. A potential focus area for the app can therefore be encouraging exercise and healthy lifestyles.

7.1.2 Advice for prevention

The individual can easily reduce the risk of falling greatly by taking certain measures. A list of measures normally recommended for older people in the target group by the authorities is given below[?]:

- Begin a regular exercise program.

- Exercises that improve balance and coordination are the most helpful.
- Is the only measure that by itself reduces the risk of falling independently of individual circumstances.
- Have your health care provider review your medicines, even over-the-counter ones.
 - Avoid medicines that can make you dizzy or sleepy.
- Have your vision checked.
- Make your home safer.
 - Remove or fasten small rugs and carpets.
 - Remove wires and cords away from commonly taken paths. Tape wires to the walls, and if necessary install an additional power outlet.
 - Keep items where you can reach them without having to climb. If you have to use a step stool, get a stable one with handrails.
 - Remove loose items that can make you trip (books, papers, clothes, etc.) from the floor and stairs.
 - Add grab bars and non-slip mats to the bathroom.
 - Make your home brighter.
 - Repair broken or uneven steps and handrails in the staircase.
 - Avoid using the staircase more than necessary, for instance by installing a light switch at the top as well as the bottom of the stairs.
 - Wear shoes, even inside. Avoid walking barefoot or wearing slippers.

7.1.3 Physiological aspects

The list below explains the physiological factors that contribute to stability. “A marked deficit in any one of these factors may be sufficient to increase the risk of falling; however, a combination of mild or moderate impairments in multiple physiological domains also may increase the risk of falling. By directly assessing an individual’s physiological abilities, intervention strategies can be implemented to target areas of deficit.” [2]

- Reaction time
 - Hand
 - Foot
- Vision
 - Contrast sensitivity
 - Visual acuity
- Vestibular function
 - Visual field dependence
- Peripheral sensation
 - Tactile sensitivity
 - Vibration sense
 - Proprioception
- Muscle force
 - Knee flexion
 - Knee extension
 - Ankle dorsiflection

There exists an array of tests that can measure the performance on these aspects. Lord et al. [2] provides one possible set of tests. Some of these are possible to implement in an app, but none of them are based on hip movement.

The test developed by Lord et al. [2] was used to classify older adults into fallers or non-fallers, and has an accuracy of 75–80% in different experiments. If a test which disregards hip movement patterns performs well, there may be reasons to believe that hip movement is only a minor factor in detecting fall risk. However,

Walking behaviour seems to be generally associated with falling. Lord et al. [3] shows that there is a negative correlation between falling and steps per minute, stride length and stride velocity. A positive correlation was shown between falling and stance duration and stance percentage. However, these physiological traits are all associated with old age, and old age is associated with falling, so the relation might be quite indirect. On the other hand, stride and stance as physiological aspects that can be measured with relative ease using a mobile phone.

7.2 Related Applications

There exist several other open source programs and commercial programs that solve related problems. These could function as inspiration for design, interfaces and functions, and open source programs can be imported and incorporated into the project. The most relevant related applications will be presented below.

7.2.1 Endomondo Sports Tracker

This is a popular app for exercising. It can be used to track routes, times and progress in training and activities. It uses GPS (and accelerometer) to find position on maps and track the location. Very easy to use, and automatically syncs up to endomondo.com. Has an inspiring design and sync-functionality.

The main part of Endomondo is the inspirational part to get people to

move; you can compare yourself to others, e.g. friends, and you have a really easy way to see what an exercise is worth in amount of calories and amount of burgers burned by an exercise.

A lot of aspects from this app can be used during design, especially labelling and representation of data.

7.2.2 GPS Status

This application is used to view detailed status about the GPS system on the phone. Has a very advanced interface with lots of numbers to show information, but for a rookie user the data rarely translates into information.

The app succeeds to use the sensors very heavily.

7.2.3 Pedometer

¹ An open source app that has a GPLv3 license that is compatible with our Apache 2.0 license. The app uses the same sensors that may be useful for the current project. Taking advantage of this could reduce development cost drastically.

The creator's description of how the step detection algorithm works²: *Basically, it aggregates the sensor values, finds the maximum and minimum, and if the difference is bigger than a value (which depends on the sensitivity setting) then it counts it as a step. There's is some additional optimization, which I arrived to through experimentation.*

7.2.4 GPS Tracker

³ This program adds the capability to store and review where you and your Android device have been. Basically you press record at the start of your trip and your phone stores the route you take. This route is drawn real-time on the Maps functionality of Android or in the background with an

¹Application can be found at <https://code.google.com/p/pedometer/>

²Taken from the FAQ at <https://github.com/bagilevi/android-pedometer>

³Application can be found at: <https://code.google.com/p/open-gpstracker>

idle device. The route is stored on your phone for review and further use. The applications tracks location by GPS, hence the name. An accurate description of the program type would be a GPS logger.

It contains the ability to log location, something that might prove useful for current project. Since it is open source and has a compatible license (GNU v3), it can be incorporated easily.

7.3 Summary of interview with medical professionals

This is a summary of the information gathered by a meeting with two medical professionals and the customer.

Present: All group members, Adjunct Associate Prof. Babak Farshchian, Prof. Jorunn L. Helbostad, Prof. Beatrice Vereijken

Time: 22. March 2013 - 14.15-15.37

Place: St.Olavs Hospital

The plan was to have a short presentation, a short GUI demonstration, and then discuss and interview with the interviewees.

Feedback on Interface and Presentation of the program

- Immediate and summarized feedback was desired, as it would be more helpful than feedback later.
- A focus on falling is not a selling point, try health or lifestyle instead
- Menu is not obvious to people without android experience
- Interface seems too complex for target groups
- Text might be too small
- Graph needs more contrast

Feedback on program functionality

- Feedback should be tailored to particular groups
- Self-testing can be useful for reducing risk and keeping awareness,
- Small tests to measure reaction time, for example reaction to light or sound
- Researchers could also be interested in data gathering
- Measuring transitions between sitting/lying and standing up
- Compare behavior on a weekly basis will give an overview of whether things are good or not

Feedback on medicinal stuff

- Preventing inactivity over time is useful to reduce risk
- People with stability problems increase risk when moving much
- Variable gait pattern is useful for predicting risk
- Step time is more robust (and hopefully easy to measure)
- A common exercise is to take a step forwards, sideways, backwards, sideways (measuring a box)
- People in risk group are those who cut out walking, taking the bus, want help from others around the house
- Irregularity in speed and length is important to look for
- Appropriate movement amount for gender and age group is usually known
- Time needed to turn in place

7.3.1 Informal usability test

During the meeting with the medical experts, the group gave an small and informal usability test to the two experts.

Prototype test

Subjects: Beatrice & Jorunn

Procedure

This test was mainly focused on how intuitive it was to navigate the application system and how easy it was to understand the data received by the system. The procedure for this was to give the test subject a phone with program installed. They would then be asked to accomplish a few short tasks, and the observers would rate how easy the subject accomplished the tasks.

Results

The data received from the system was easy to spot and easy to understand. The subjects had some difficulty finding the menu button for the first time, but had no problems with using it later on. It was overall relatively easy to navigate the system. However, the complexity was likely to be a bit much for the target group.

7.4 Lessons learned

The research provided significant benefits for the group by providing a solid knowledge base for the application domain, so time spent on research was well spent. However, in retrospect the group was able to identify some faults in their research planning, and learned some lessons about how to avoid the same mistakes in the future:

- Even though the research were accurate, the group was reluctant to employ the research results in the application before they were confirmed by the health experts. As the meeting with the health experts did not

take place before 22.03, the group lost valuable time waiting, time that could have been spent for implementation. The lesson learned is to rely on research results, at least until a more reliable source of information is available (in this project, the expert group).

- Even though research was useful in the domains that were covered, some areas that could have benefited from research were not identified during planning, and were therefore not given thorough research. Because of this, some of the customer's demands were misinterpreted at first, forcing the group to change the application architecture at a later stage. In particular, the group was unfamiliar with the standards of Android development, especially the general architecture of Content Provider. A solid foundation in this area could have aided understanding. The lesson learned is to spend more time trying to identify potential research areas.

Bibliography

- [1] "Falls in older people: epidemiology, risk factors and strategies for prevention". Laurence Z. Rubenstein. *Age and Ageing* 2006; 35-S2:ii37-ii41.
- [2] "A physiological Profile Approach to Falls Risk Assessment and Prevention", Stephen R. Lord, Hylton B. Menz and Anne Tiedemann, *Journal of the American Physical Therapy Association*.
- [3] "Sensori-motor Function, Gait Patterns and Falls in Community-dwelling Women". Stephen R. Lord, David G. Lloyd, Sek Keung Li. *Age and Ageing* 1996;25:292-299.
- [4] "What you can do to prevent falls", Centers for Disease Control and Prevention.
- [5] "Preventing Falls: How to Develop Community-based Fall Prevention Programs for Older Adults", Centers for Disease Control and Prevention.
- [6] "Muscle strength, activity, housing and the risk of falls in elderly people." Wickham C et al., *Age and ageing*, 1989, 18:47-51
- [7] "What are the main risk factors for falls among older people and what are the most effective interventions to prevent these falls?" Todd C, Skelton D. (2004) Copenhagen, WHO Regional Office for Europe

Appendix A

Appendix

Appendix content goes here.

A.1 Previous WBS

This is where the Work Breakdown Structures from early development is placed here with the most recent placed earliest, as can be seen in Figures A.1 and A.2.

A.2 Reports

The work done was described as short periods called sprints.

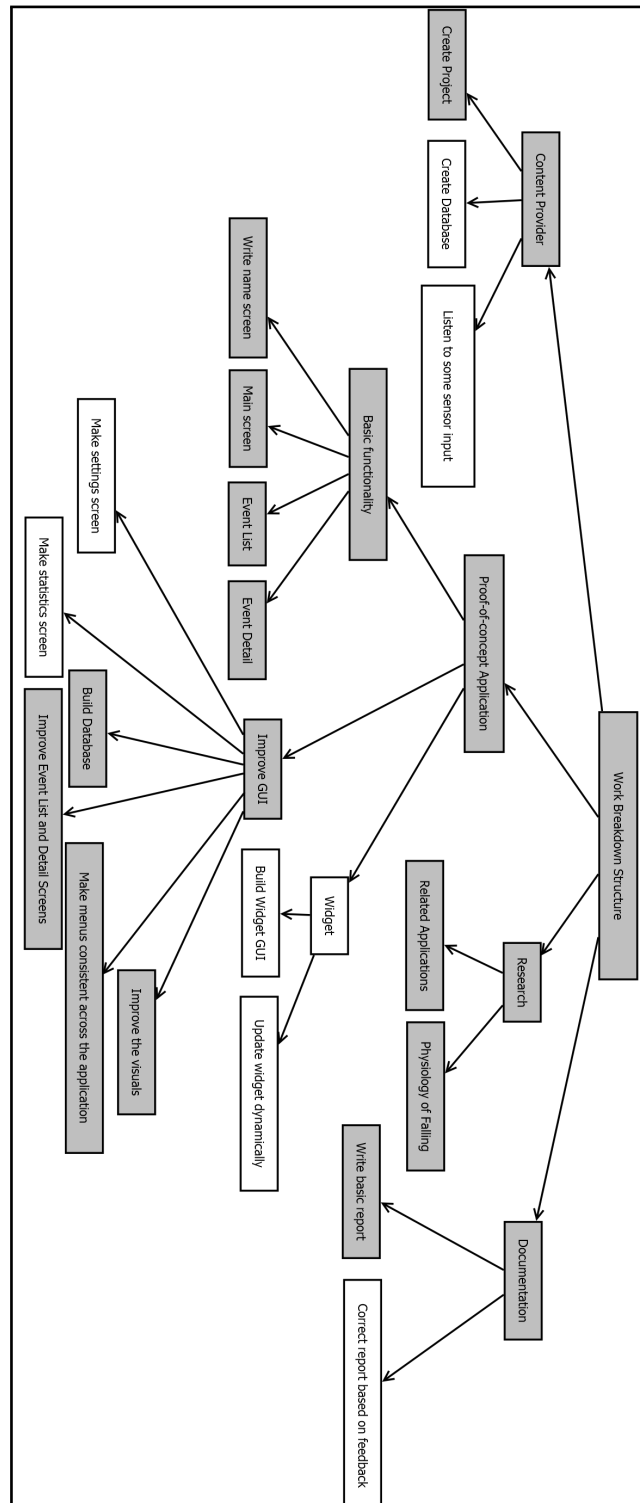


Figure A.1: The WBS as per 8.3.13

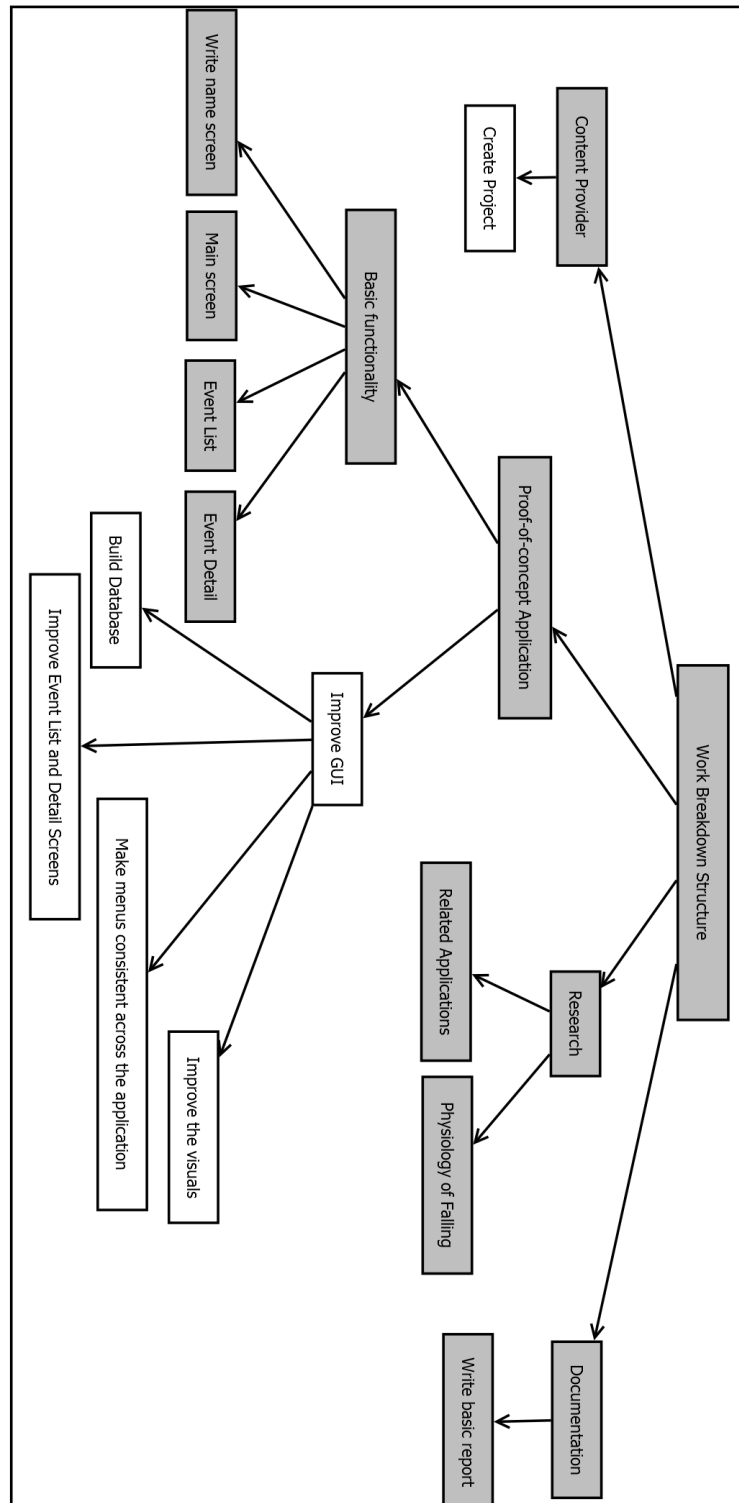


Figure A.2: The WBS as per 1.3.13

Figure A.3: Short summary of work done sorted by sprint

Sprint nr.	Date	Summary
Sprint 1:	03.02.13 - 08.02.13	Developing user stories and paper prototypes of the GUI.
Sprint 2:	08.02.13 - 15.02.13	Developing a mock-up application demonstrating the GUI.
Sprint 3:	15.02.13 - 22.02.13	Improving UI and functionality for the prototype, researching medical factors.
Sprint 4:	22.02.13 - 01.03.13	Improving secondary functionality for the prototype (settings, statistics, relatives screens),researching content-providers and alternative solutions.
Sprint 5:	01.03.13 - 07.03.13	Creating content-provider, creating presentation, finishing secondary screens (as described above), including pedometer, writing test-cases.
Sprint 6:	08.03.13 - 15.03.13	
Sprint 7:	16.03.13 - 22.03.13	todo
Sprint 8:	08.03.13 - 16.03.13	todo
Easter	-	little work done
Sprint 9:	03.04.13 - 5.04.13	Changing plans, removing pedometer app from project, planning new project structure
Sprint 10:	6.04.13 - 12.04.13	Creating own content-provider and step-detector, integrating all three components
Sprint 11:	13.04.13 - 19.04.13	Fixing documentation according to feedback

A.3 Apache license and notice

Apache License

Version 2.0, January 2004

<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work

(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a

cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by

reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

A.4 Meeting summaries

Regular status reports was a part of necessary documentation. Of particular importance was reports to the supervisor, and reports done to the other group members. This status report was written during sprint 4. The reports were included in chronological order.

Project FallPrevention Group 08

Status report for week 09

1: Progress summary

Prototype:

- The mock-up application has been developed further, in particular with a better-looking GUI and several more screens to display data.
- A database has been build with the application to store all the required data.
- Enabled localization for the application, and included Norwegian and English text. File for the german version are available, but not yet translated.

Research:

- The group has learnt about ContentProviders, which Android uses to encapsulate data from outside of the application.
- A short summary on consequences of, and what factors results in, falling among the elderly, has been written and read by the entire group.

2: Open/Closed problems

- **Group members out of town:** Some group members had trips out-of-town, resulting in less work being done. This was solved by not planning more work to be done in those time frames, than what the remainder of the group could accomplish. With group members remembering to inform the rest of the group in advance, it does seem like this problem is closed.
- **Customer taking vacation:** The group had to skip one meeting with the customer, as the customer took some time off for vacation. As the product is still in a very primitive state, and any further work need to be discussed with the customer, this caused a delay of almost a week, as there was limited work to do in the meantime.

3: Planned activity for next period

The plans for this week are as follows:

- With the graphical and persistence-related parts of the prototype mostly implemented, the details and implementation of the ContentProvider handling interaction between the application and the motion detector seems like it will need more work.
- Complete the implementation of the GUI, and polish it by making it more readable, and adding the rest of the icons necessary to display all the intended permutations.
- Prepare a demonstration for the experts the customer will put the group in touch with, as the expert is not experienced with application development and might need a more user-friendly demonstration.
- Research applications that use sensor technology.

The plans for the period beginning this friday will need to be discussed with the customer and any medical expert the group has yet to meet.

4: Updated risks analysis

While we will not discuss updating the risk analysis until the meeting this wednesday, it is evident that some risks need to be updated:

- “Customer not able to meet with the group” will need to be added to the risk list, although it does not seem very likely to happen more than once. As the group has some understanding of what parts of the project needs working on, it will not become a large delay unless it happens several times in a row.
- “Computer problems” has had a larger influence than we have expected.
- A new risk, “participant not able to work as much as required” will need to be added.

Project FallPrevention group 08

Status report for week 11

1:Progress summary

Prototype:

- Content Provider implemented, but not yet integrated with prototype application.
- Content Provider now has database functionality.
- Integrated pedometer functions
- Application can now add contacts for the purpose of automatically contacting the contact in particular circumstances.
- Application now has settings screen with a few functions.

Research:

- Research into Android applications Edomondo Sports Tracker, Pedometer, and GPS Status has been done, as these apps use and display sensor data in ways that the group might find very useful.
- Pedometer is also open source and has an compatible licence, so it has been integrated in our project and application
- Made presentation and prepared questions for meeting with medical experts.

Documentation:

- Feedback from the preliminary report has been fixed
- Requirements partially updated to current needs
- Testing plan researched and prepared

2: Open/Closed problems

There has mainly been three problems since last report. They are: Unpredictable illness for group members, e-mail still being inefficient at communicating and getting responses swiftly, and group member not being equally experienced with the tools being used, slowing down productivity.

3:Planned activity for next period

- Finish filling testing plan with tests
- Update requirements in the report
- Update architecture in the report to reflect the current state of the application
- Have meeting with medical experts, and gather information about what the application should focus on.
- Integrate Pedometer into the Content-Provider, and get the application ready to detect
- Take a few passes with refactoring and documenting over all the code, with an aim to improve readability and quality of code.

4: Updated risks analysis

The risks as described above is already included in the risk list. With the topmost item on the list being implementation difficulties(mainly people not having the programming competencies the tasks demand). Documenting and refactoring the code, something which everyone will take part in, will likely improve the competency of the group, and their understanding of the code, to such a level that the likelihood of this problem will be reduced.