Valens - An Android application system to detect fall risk among the elderly
Group 8

Vassdal, Johannes Willumsen
johannes.vassdal@gmail.com

Aamot, Elias
eliasaa@stud.ntnu.no

Tomren, Filip Andre Larsen
filip@tomren.it

Pham, Dat-Danny
datdanny@hotmail.com

Kücükyareli,Tayfun
tayfun.kucukyareli@live.de

May 27, 2013

Contents

1 Introduction **5**

1.1 IT2901 5

1.2 Group introduction 6

1.3 Stakeholder introduction 6

1.3.1 SINTEF - Customer 6

1.3.2 Target audience 7

1.4 Interested parties 7

1.4.1 Medical Experts 7

1.4.2 Supervisor - Tinna Tómasdóttir 8

2 Requirements **9**

2.1 Initial requirements 9

2.2 Elaborating on the requirements 10

2.2.1 Functional requirements 10

2.2.2 Non-functional requirements 10

2.3 Omitted requirements 11

2.4 License 12

3 Prestudies and Alternate Solutions **14**

3.1 Causes, consequences and how to avoid falling 15

3.1.1 Risk factors 15

3.1.2 Advice for prevention 16

3.1.3 Physiological aspects 18

3.2 Related Applications 19

3.2.1 Endomondo Sports Tracker 19

3.2.2 GPS Status 20
3.2.3 Pedometer 20
3.2.4 GPS Tracker 20

4 Software development and management processes 22
4.1 Terms 22
4.2 Development process 24
4.3 Meetings with customer and supervisor 26
4.4 Plans 27
4.5 Team Roles and Organization 28
4.6 Work Breakdown Structure 29

5 Tools 31
5.1 Github 31
5.2 Android Developer Tools Bundle 31
5.3 Trello 32
5.4 itslearning 32
5.5 Email 34
5.6 LaTeX 34

6 Design and Architecture 35
6.1 Overall Architecture 35
6.2 Architecture of Valens Health Helper 36
6.3 Architecture of the Content Provider 42
6.3.1 Data model 43
6.4 Architecture of the Valens Step Detector 43
6.5 Class description and diagram 46
6.5.1 Translations 46

7 Step Detection Algorithm 51
7.1 Development and purpose of Algorithm 51
7.2 Core Algorithm 51
7.3 Application of the algorithm to live data 53
7.4 Constant values 55

7.5 Gait calculations . . . 55
7.6 Room for improvement . . . 56

8 Testing **58**
8.1 Levels, methods and types . . . 58
8.1.1 Levels . . . 58
8.1.2 Methods . . . 59
8.1.3 Types . . . 60
8.2 Plans and process . . . 61
8.2.1 Unit testing . . . 61
8.2.2 Inspections . . . 61
8.2.3 Integration testing . . . 61
8.2.4 System testing . . . 62
8.2.5 Acceptance testing . . . 63
8.2.6 Informal usability test . . . 63
8.2.7 Acceptance testing and deployment . . . 64
8.3 Tests used under development . . . 64
8.3.1 GUI Tests . . . 64

9 Conclusion **67**
9.1 Lessons learned . . . 67
9.1.1 New Experiences . . . 67
9.1.2 Learning experiences from development and tool use . . . 67
9.1.3 Lessons learned about Research and Prestudies . . . 68
9.2 Conflict handling . . . 69
9.3 Further work . . . 69

Appendix A WBS **73**
A.1 Previous WBS . . . 73

Appendix B Work summary **81**
B.1 Sprint summaries . . . 81
B.2 Meeting summaries . . . 85

Appendix C Test results **90**

C.1 Summary of interview with medical professionals 93

Appendix D Time-sheet **95**

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 a 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 fall-related accidents.

This project is a cooperation between SINTEF and a group of students at NTNU, as a part of the European research project FARSEEING. The students participate as a part of the course IT2901 - Informatics Project II.

The general goal of the project is to explore potential uses for smart phones in fall risk assessment and prevention. Specifically, the goal was to develop a model for fall risk level based on movement detected by common sensors found in Android smart phones, and build an API to make the model accessible for third-party developers.

1.1 IT2901

IT2901 - Informatics Project II is a course given at NTNU, where a group of students are given a project to work on independently over the course of a semester. The projects are provided by private and public enterprises in cooperation with NTNU. The university provides a supervisor to give the students feedback throughout the semester, but the students themselves are

responsible for direct communication with the customer. The course is worth 15 ECTS, which amounts to approximately 20 hours of work per week for each student.

1.2 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 no experience with any Android application development.

1.3 Stakeholder introduction

In addition to the developers, the following parties were stakeholders in the project.

1.3.1 SINTEF - Customer

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.

FARSEEING

"FARSEEING is a collaborative European Commission funded research project with 11 partners distributed in 7 EU countries. It aims to provide a thematic network focusing on the issue of promoting healthy, independent living for older adults. FARSEEING aims to promote better prediction and prevention of falls and to support older adults with a focus on ICT devices and the unique proactive opportunities they can provide to older adults to support

them in their own environment".[1]

1.3.2 Target audience

The target audience for the system are:

- Elderly that are self-sufficient and relatively healthy, but have some concerns regarding their own stability. They benefit from the application as it provides feedback about their risk level.
- Health personnel. The application can work as an indicator for fall risk among the patients, and perhaps provide insights into possible solutions and causes to the falling problem.
- Concerned relatives.

One major problem that was pointed out by the health experts is that the first target audience does not, in fact, exist. Falling is an enormous taboo, and even people with a history of falling tend to convince themselves that every fall is an individual exception. To cope with this, the application has to be marketed as a general health promoter when targeting the elderly, but run fall risk detection in the background. Targeting the other two audience groups, the application can still be marketed as originally planned.

1.4 Interested parties

Groups who are not stakeholders, but still have been of great help.

1.4.1 Medical Experts

Profs Jorun L. Helbostad and Beatrix Vereijken provided invaluable help with understanding questions from the medical domain. They were also associated with the FARSEEING project.

[1] Quote from the "About Us" section of the FARSEEING website `http://farseeingresearch.eu`

1.4.2 Supervisor - Tinna Tómasdóttir

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. She took part in evaluating the project.

Chapter 2

Requirements

The main goal of the project is to explore the advanced technology in any standard smartphone. A standard smartphone is packed with heavy duty processors and sensors. To put these senors and processing power this project focus on developing a prototype app for Android Operating System to predict risk of falling.

2.1 Initial requirements

The initial requirements originated largely from the customer, with some course requirements added in. The requirements presented by the customer at the beginning of the project can be summarized as follows:

- A model for fall risk assessment based on sensors in Android phones should be developed.
- The model should be accessible through a Content Provider.
- The model should be accompanied by a proof-of-concept demo application.
- The product should be available as open source through the Apache 2.0 license, explained below.
- Development should follow an incremental approach.

2.2 Elaborating on the requirements

The initial requirements were too vague to base an implementation on, so a requirements elicitation process was initiated, where the group in conjunction with the customer, and later the health experts, gradually elaborated upon the requirements. The process resulted in a list of more precise requirement, as presented below:

2.2.1 Functional requirements

- A standard Android content provider should provide an interface to the generated risk data. The data model should contain enough details to be able give at least a basic predication of fall risk. The data model should include

  Gait Speed is the interval between two steps, and measuring it is useful, as it can be used to measure Risk for falling and as a shorthand for physical ability.

  Gait variability is the standard deviation for gait speed over time. This is significant, because a large variability is correlated with increased risk for falling.

  Step Count is number of steps done over a period of time.

- An example application should the developed that demonstrates the data model.

2.2.2 Non-functional requirements

- Development should be done in short iterations, and results should be presentable at each weekly customer meeting.

- The demo application should appear as a health assistant, not as an application for fall risk assessment, as the elderly are reluctant to admit they have a risk for falling, and would not use such an application.

- The demo application should be easy to use, even for elderly people unfamiliar with electronic equipment. This includes localization support at least for Norwegian, few visual details, and simple feedback.
- The demo application should provide more detailed information that is useful for health personnel without a computational background.
- The application system should be modular and well-documented, so that it is easy to implement new applications based on the content provider, as well as connect new data sources to the content provider.

2.3 Omitted requirements

The customer did usually only specify requirements that should be done for the current sprint, but the group were given a task to make some user stories. The group then formed some requirements from the user stories, which turned out to be too extensive given the time limit.

Contacts and notifications An initially desired function was to maintain a list of contacts and their contact information and the possibility to notify the contacts by e-mail or SMS, in order to accommodate the second target group. This requirement was excluded later on because it was difficult to determine which types of situations that should prompt notifications, and that it would require significant amounts of time to develop this non-essential functionality. Before the decision was made to exclude this functionality, a system for creating, importing and maintaining a list of contacts was developed, and exists in the source code, but the related screens are not accessible in the application.

Stride length In addition to step count and gait parameters, *stride length* is another predicator of fall risk that may be measurable using sensors that are available on Android. One approach could be to employ a location detector (for instance GPS) to track distance walked, and find step length as

$$\frac{stepsTaken}{distanceWalked}$$

However, the additional effort required to implement location awareness overshadowed the perceived predicative power of the additional predicator.

Inactivity detection Long periods of inactivity are bad for the general health, as well as strong indicators of fall risk. While some effort was put into studying the patterns related to sitting down and getting up to be used in the context of inactivity detection, time constraints and the need to focus on step detection forces this requirement to be excluded.

2.4 License

The results of the project were to be released under the Apache 2.0 license, as requested from the customer. It is an open source license, which can be found at Apache's web site.[1] Highlights of what can, cannot and must be done under the license is described here:

Must be done

- Include copyright
- Include the license
- State what changes have 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

[1] `http://www.apache.org/licenses/LICENSE-2.0.html`

- Trademark the licensed work

The specific terms can be found at the mentioned address, along with License and Notice files.

Chapter 3

Prestudies and Alternate Solutions

During the planning stage, the group 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 itself (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 are reproduced in the following sections.

3.1 Causes, consequences and how to avoid falling

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

3.1.1 Risk factors

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

- **Biological risk factors (sorted by order of significance): [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 [2]
  - Medications [2]
  - Alcohol use [2]
  - Living alone [3]
- **Home/environmental risk factors (not in sorted order): [4]**
  - 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 are involved in only 30-50% of all falls[1, 2]. 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 one of the project goals is to develop a model utilizing smart-phone sensors, it might be necessary to down-prioritize identifying risk for dizziness/vertigo and drop attacks, as well as ignore external factors. Instead it seems beneficial to focus purely on physiological aspects, such as gait or balance, which are easier to measure.

Many older adults are unaware of their risk factors, and therefore unable to take preventive action. 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 very beneficial, especially when the results are discussed with a healthcare provider[2]. This fact illustrates the usefulness of the planned application.

It has been shown that many of the biological risk factors can be reduced effectively by preforming regular physical activities. Specifically strength, gait and balance has been shown to be improvable by simple exercise regimes - even for frail patients - in a number of studies[2, 4, 5]. A potential focus area for the app can therefore be encouraging exercise and healthy lifestyles.

3.1.2 Advice for prevention

The individual can easily reduce the risk of falling greatly by taking certain measures. A list of measures normally, given by the authorities, recommended for older people in the risk group is given below[6]:

- 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.

3.1.3 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." [5]

- 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. provides one possible set of tests. Some of these are possible to implement in an app, but none of them are based purely on hip movement.

The test developed by Lord et al. was used to classify older adults into fallers or non-fallers, and has an accuracy of $75 - 80\%$ in different experiments.[5] 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.

Walking behaviour seems to be generally associated with falling. Lord et al. 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]

3.2 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 are presented below.

3.2.1 Endomondo Sports Tracker[1]

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.

[1]This application can be found at `http://www.endomondo.com`

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 labeling and representation of data.

3.2.2 GPS Status [2]

This application is used to view detailed status about the GPS system on the phone. It 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.

3.2.3 Pedometer[3]

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 works: *"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."*[4]

3.2.4 GPS Tracker[5]

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

[2]This application can be found at `https://play.google.com/store/apps/details?id=com.eclipsim.gpsstatus2`

[3]Application can be found at `https://code.google.com/p/pedometer/`

[4]Taken from the FAQ at `https://github.com/bagilevi/android-pedometer`

[5]Application can be found at: `https://code.google.com/p/open-gpstracker`

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 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.

Chapter 4

Software development and management processes

The following chapter explains the software development and management processes that were employed throughout the project. In the following section some of the necessary terminology will be introduced and explained. Subsequently the process description will be given, followed by a comparison to established process models.

4.1 Terms

Here follows a description of terms as used in the current project.

Agile methods are a set of software development methods that emphasize on adapting to change over following a predetermined plan, continuous customer collaboration over contract negotiation and implementation over documentation.

Extreme Programming (XP) is an agile software development methodology which favors improving the code and quick responses to customer desires. The customer is involved with the process continuously, and keeps giving feedback on the most recent progress. Releases of new code is also supposed to happen often, so both the customer and the

developers are aware what the current progress is, and can give feedback on it. Both the code and requirements are expected to change during the development process. It is incremental in that what the customer wants is implemented swiftly, and stays as a part of the code. It is iterative in that code that is added will likely be subject to change and refactoring during the development process. Constant refactoring and pair-programming is made use of so the quality of the code will be kept as high as possible, as it will be difficult to plan and design for code-quality before production of code starts.

Iterative development is an approach to software development that is "based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed"[1]. Benefits to iterative development include adaptability to changing requirement and facilitation of feedback. In the negative side, it can be hard to keep an overview of the progress of the entire system, making it harder to plan ahead for deadlines.

Pair programming (PP) is a programming technique taken from XP. In PP, coders work in pairs that work on the same workstation. As each line of code is looked at by at least two people, code development simultaneously acts as a review process, making code developed by a programmer pair more reliable and readable than code developed by an individual programmer. For complex coding tasks, the productivity of a programmer pair is normally higher than that of two individual programmers, considering the reduced effort required for bug fixing. For routine tasks, however, productivity is lower in a programming pair.

Plan-driven development is an approach to software development that recommends careful planning of the software development process, work breakdown and scheduling before starting the implementation process.

[1]Sommerville, p32 [8]

Sprint is the period unit in which work is planned and done. It is customary to operate with sprints of 2-4 weeks, but the current project operated with sprints of 1-2 weeks.

Spike is a sprint spent primarily or solely on non-programming activities, such as research, documentation or planning.

Stand-up is a meeting in which all the group members explain shortly what they have been doing since the last meeting, what they are planning to do until next time, and any help or advice they need to continue. Stand-ups function as a binding factor that keeps the entire group on the same page, and provides flexibility by giving the opportunity to change the plans and tasks partway through the sprint.

4.2 Development process

The customer required an incremental development process where, ideally, a new version of the product could be demonstrated at every weekly customer meeting. Furthermore, plans for the following week were to be sketched in collaboration with the customer at the weekly meeting. With such short-term plans, and as the requirements were still vague and the product to be developed largely experimental, a high degree of flexibility was required. A plan-driven approach was therefore impossible at the beginning of the project. The process model employed in the early stages of development was therefore based on agile methods, and extreme programming in particular. Due to the customer requirements, sprint length was set to one week, with the possibility for two week sprints in certain cases, such as when the customer is unable to show up to a weekly meeting, or the required tasks are more extensive than usual.

Meetings between the customer and the group normally took place on Fridays, and were followed by a group planning and work session. Stand-ups were originally planned for Wednesdays and Sundays, as these were the only times all the group members were available at the same time. The meetings at Sundays were conducted online, and the Wednesday meetings

at the university. The Sunday stand-ups were abolished after a few weeks, as little work had normally been done between the Friday session and the Sunday stand-up. To compensate for the relative low frequency of stand-ups (Agile development teams in full-time work environments normally hold daily stand-ups.) other communication channels, such as e-mail and Trello[2] were heavily used.

The process model incorporated a liberal approach to pair-programming, where coders formed programming pairs only for tasks deemed complex. Another principle that was taken from XP was that of collective ownership of the code. Emphasizing the fact that nobody "owns" a particular class made it easier for the group members to accept changes when demanded, thus ensuring development agility[3].

The task assignment procedure was largely agile: After the weekly sprint planning meeting, the individual group members could choose which tasks they would like to work on in the following sprint. Tasks were thus assigned in a first-come first-served fashion. When a member finished one task, a new one would be taken. The Group Leader only had to assign tasks in exceptional cases.

Some elements of extreme programming were not employed in the development process, particularly when it comes to testing. XP advocates *Test-Driven Development* (TDD), where automated unit tests[4] are developed before the corresponding code is written. As no group members had any experience in test automation or TDD, employing such an approach appeared to entail more challenges than benefits.

After Easter several changes in the circumstances prompted a change in development methodology. These included:

- A much clearer definition of the requirements and the project goals had gradually emerged.

[2]See page 32 for more information

[3]In the Javadocs, every class has a listed author that can be contacted if any future developer has any questions regarding the code in that class. While this may break the principle of collective ownership, the "author" of a class selected after development were finished, so as to not interfere with the development process.

[4]See the testing chapter for details on unit testing

- The time left until the final deadline had decreased significantly.
- The number of potential tasks left had remained close to constant throughout the semester, as finishing one task often leads to insights as to what can be done further.
- The group had attained a better understanding of its own work process.
- Significant integration and system testing was needed.

These circumstances revealed the necessity for as well as the possibility to make concrete plans, and prioritize between the potential tasks. Therefore, the development model was changed to a plan-driven approach that still enforced incremental delivery. Some flexibility was still maintained, by making a buffer of optional tasks in case of delays or unforeseen requirement changes. Useful elements from XP were still preserved, such as pair programming and individual choice of tasks.

4.3 Meetings with customer and supervisor

Weekly meetings with the customer took place every Friday, as explained above. Also, meetings with the supervisor were conducted every second week.

The meetings had several positive effects:

- The project stakeholders were kept up-to-date.
- The progress of the project was predictable.
- The developers got feedback before and after the tasks are done.
- The project was split in small pieces of actionable and specific smaller tasks.
- These smaller elements were discussed and considered before they were started.

- Due to the smaller pieces of progress, mistakes were backtracked with a minimum of wasted work.
- Customer got to participate in the development.
- The group could take action when the supervisor or customer gave feedback on the results.

4.4 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 Appendix B.1, and provided a short summary of what was accomplished in a particular time-frame.

The customer favoured a user-oriented development method which focused on first developing a complete mock-up GUI, and base development of data models and back-end functionality on what the GUI requires to function properly. Development of the back-end model should in this method proceed "top-down", in that functionality that directly influences user experience should be implemented before more fundamental, data-manipulating methods. Advocates of this development method claim that first developing the GUI puts user-friendliness in focus, and constrains which functions are required in the back-end, so that no superfluous methods need to be implemented. On the downside, this may cause some of the back-end methods to be down-prioritized due to time constraints.

A radically different approach would be a "bottom-up" method that focuses on first developing the most challenging and novel parts of the system, in particular algorithms for step detection and calculation of fall risk scores. This method would only start developing increasingly more user-oriented functionality when the more fundamental methods have been verified. An advantage of this method is that it allocates all the time necessary to master the complex issues first. On the other hand, this could cause development of the user-oriented features to be neglected due to time constraints.

In fact, the difference between these two approaches only reveals itself in the product when time constraints force certain features to be down-prioritized. Implementation and verification of fall risk assessment features[5] was one of them due to this project followed a user-oriented development method. GUI development got a lot more attention in accordance to the customers wishes. If a "bottom-up" method had been used, it seems likely that the project would have resulted in well-tested and extensive fall risk assessment components, but with little or no GUI intended for the target group.

4.5 Team Roles and Organization

Extreme programming advocates a flat structure, where every member of the group were considered to be capable and willing to work at all the tasks, and as such there was not much place for specific roles among the group. However, to make sure that no area was neglected, some sort of overview was required, and three roles were therefore defined. Common for the roles is that they entail responsibilities for a particular domain, but provide no advantages.

Group Leader Elias. Responsible for making sure that no areas are neglected, and that the project has maintains a momentum of progress.

Document-organizer Johannes. Because the report is a major task, but easily neglected during development, somebody was required to emphasize the importance of continually working on the report. Also responsible for organizing documentation and keeping track of required reports.

Customer Contact Filip. To avoid sending redundant or contradictory messages to the customer, one person was assigned to handle customer communication.

[5] Acceptance testing confirmed that the users shared this impression, see section 8.2.5 for details.

Otherwise the group had no codified responsibilities.

4.6 Work Breakdown Structure

With an agile, incremental development methodology, work breakdown was only possible within the individual sprints, based on the short-term customer requirements as well as on the state of the application at that time. In other words, a WBS was developed incrementally by adjoining the new WBS to the previous. After the change to a plan-based development model, the work breakdown structure (WBS) for the remainder of the project was developed. A graphical representation of the final WBS can be seen in Figure 4.1.

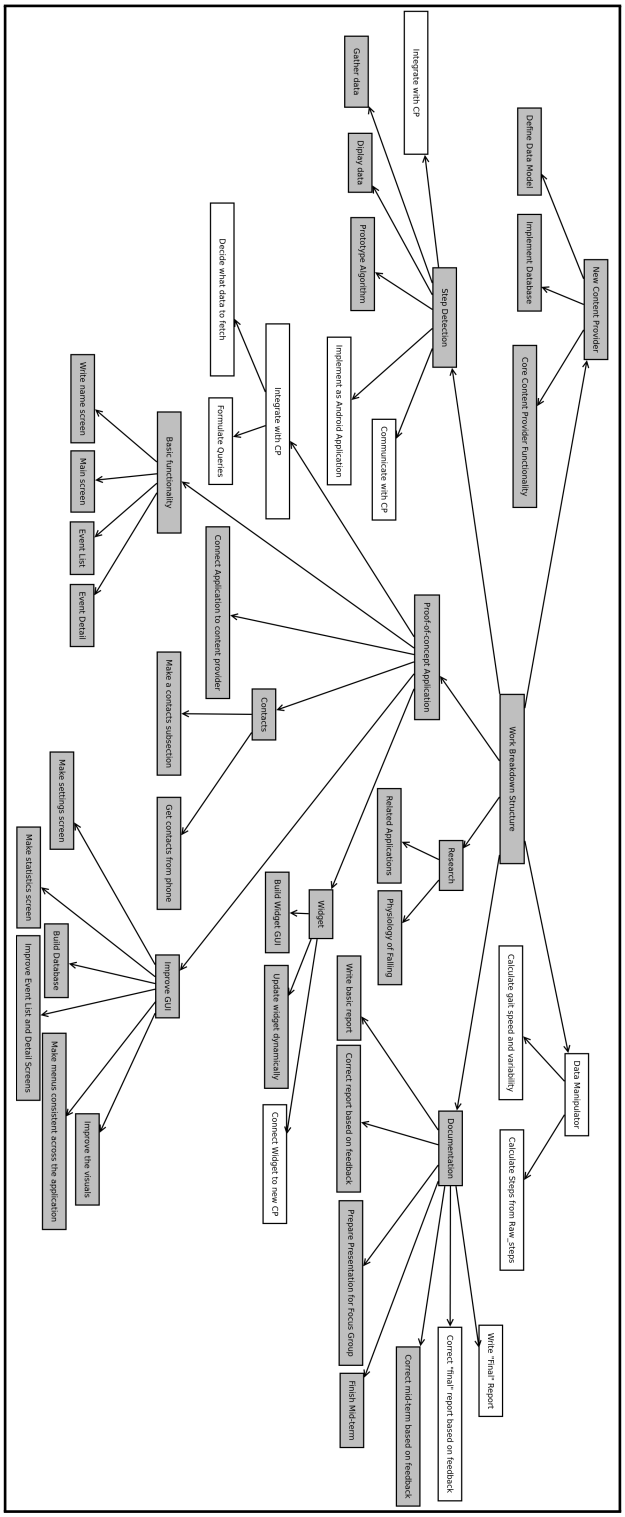


Figure 4.1: The final WBS

Chapter 5

Tools

5.1 Github

It was requested by the customer that the group should use a version control system, with Github being preferable, to share code and perform version control. It is a system used to collaborate on writing code, do version control, and share the code with others. Github was used to share and synchronize 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. Github is mostly used to update the followers of a repository and give them the newest version of the code, so it is very ideal for coding in groups, where its users can simply push their finished work and the repository will automatically merge it with the already existing code, if done correctly. Github has browser-based interfaces, download-able clients, and a robust command-line interface, meaning all the members of the group could make use of it. There were some problems associated with learning how to use and fix problems with it, as not all group members had experience with this tool, though other group members had heavy previous experience from other courses. It allowed the group to work on the same project from multiple computers, without risking overwriting the work done by the other developers. In the case of new code not working, it was also simple to roll back to a earlier version of the code.

5.2 Android Developer Tools Bundle

An Integrated Development Environment(IDE) that fulfilled the following requirements was needed:

- Had to integrate support for Android Application Development.
- Was able fairly easy to learn.

The group decided to go with the Android Developer Tools Bundle, which essentially provided us with the Eclipse IDE and the Android Developer Tools. This bundle included everything needed to start developing an Android Application:

- Eclipse IDE + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

5.3 Trello

Trello is a online collaboration tool with the ability to create interactive online boards. This helps any projects and groups to organize what is being worked on, who is working on what, and where in the progress some task is. Trello quickly became the most valuable tool to track the project tasks and progress, and to know who was working on which tasks. It simplified the group leaders task of maintaining control over project tasks a lot easier than it would have been without it. Shown in Figure 5.1 and Figure 5.2 is an example of the Trello board, and a card representing a single task and sub-tasks. The board shows tasks sorted in several columns, according to priority and status. The card also includes information on which developers are assigned to the tasks, and how far the task has progressed.

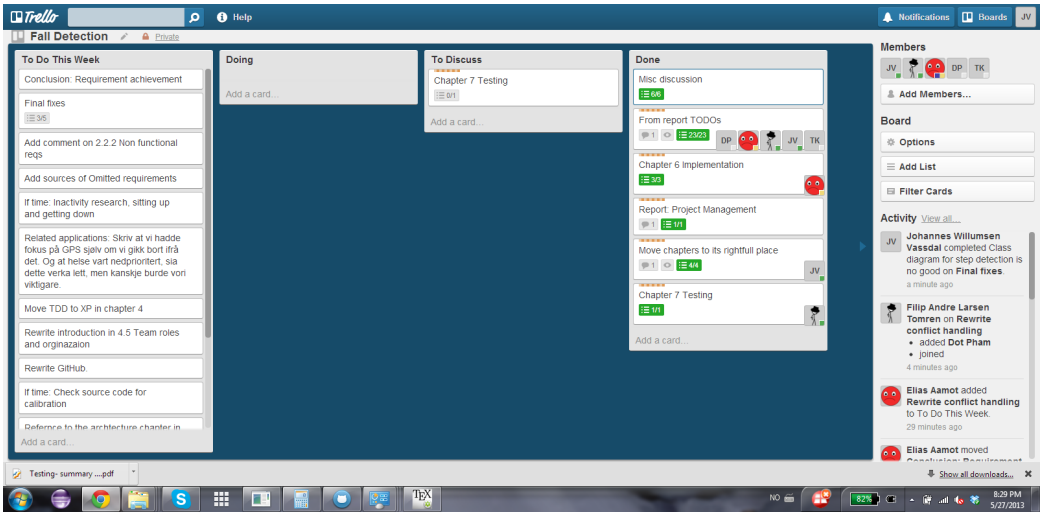


Figure 5.1: The Trello board

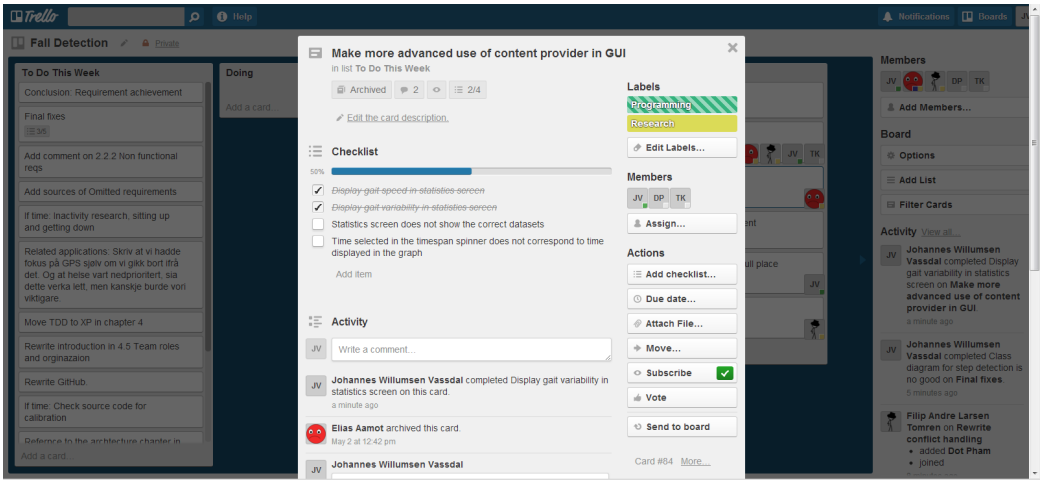


Figure 5.2: Details on a Trello card, including checklist and people assigned

5.4 itslearning

Itslearning was used to distribute information that was not time -critical, with a message board being used. Itslearning 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. This also lead to it not being used as much as it should be the group, and its use quickly vanished.

5.5 Email

Email was used for time critical communication, and for information that needed feedback swiftly, often within the same day. It also became our main communication channel to distribute messages between members of the group.

5.6 LaTeX

LaTeX was used to write this report. LaTeX is a typesetting language, with support for varied formatting, including images and including other documents. It also has the option allowed for inclusion of multiple LaTeX documents (.tex documents). This leads to improved readability of LaTeX code and increased 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. The editor used was Texmaker, an free latex-editor available on multiple platforms.

Chapter 6

Design and Architecture

6.1 Overall Architecture

One of the customer requirements was that a Content Provider be developed to provide access to the model for movement to be designed. Content providers function as data repositories that enable other applications to interact with the stored data. More specifically, content providers provide CRUD (create, retrieve, update, delete) functions. Content providers are intended to be used with a repository architecture. In a repository architecture, there is a central data repository which maintains the shared data between the components. The components only communicate directly with the repository, and are therefore independent of each other. This independence makes it easy to add or remove components from the application system, which was emphasized as a major requirement from the customer. While it is possible to embed a content provider within a single application, this causes the independence to be lost, nullifying the advantage of using a content provider in the first place.

The system that has been designed according to the repository architecture, and consists of five component apps, each of these components fulfills a well-defined role in the application system architecture:

Valens Content Provider has the role of the repository in the repository architecture.

Valens Health Helper 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.

Valens Step Detector the sensor model. The sensor model listens to the sensor data, and feeds the content provider with the timestamps of any detected steps.

Valens Content Feeder a service of deriving secondary data from the raw data provided by the step detector(s). This includes filtering out steps that have been counted by multiple step detectors to ensure that every step is counted just once, as well as deriving gait parameters. The code is integrated with the Content Providers APK for reasons of maintainability, but they are still logically independent.

6.2 Architecture of Valens Health Helper

The application follows standard android application architecture, which is based on a Model-View-Controller (MVC) architecture. In a MVC architecture, which is a fairly common architecture for GUI-based applications, there are three components:

The model contains the data and logic of the system.

The view is a visual representation of the data.

The controller receives input and converts it into commands for the model or view.

In Android, the view is described using XML layout files, the controller is defined using activity classes and the model is defined using the remaining Java classes. In practice, however, the android architecture does not separate the MVC components clearly. For instance, certain graphical features have to be implemented using the activity classes. Also, not a lot of effort was put into maintaining a pure MVC architecture.

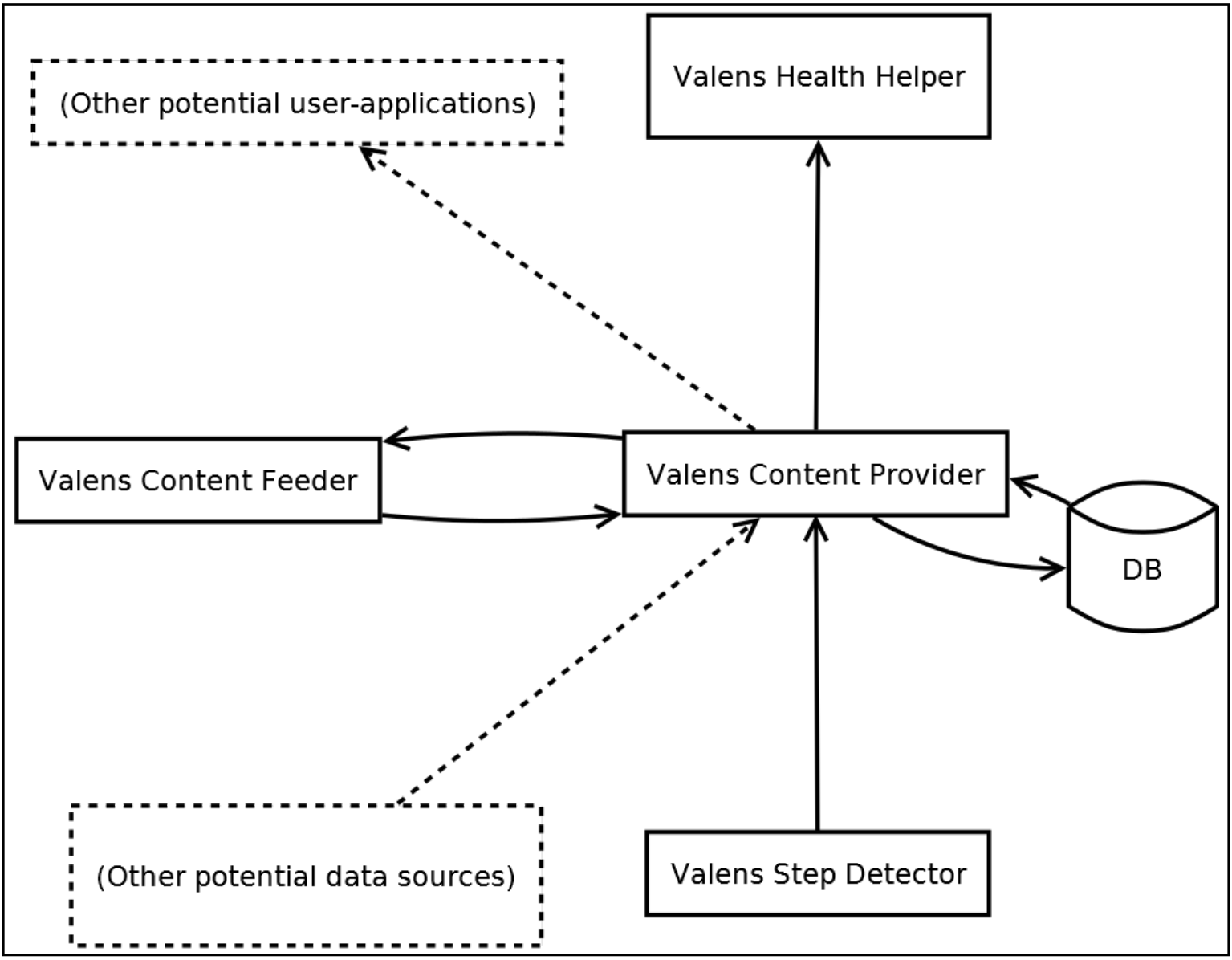


Figure 6.1: The over-all system architecture

Furthermore, in Android, an additional view layer separates the actual string values and other resources from the layout definitions. Resources are placed in a separate folder and files. The separation of layout and strings enables different localizations, so that the application can provide a user interface in different languages easily.

In the android architecture, 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.

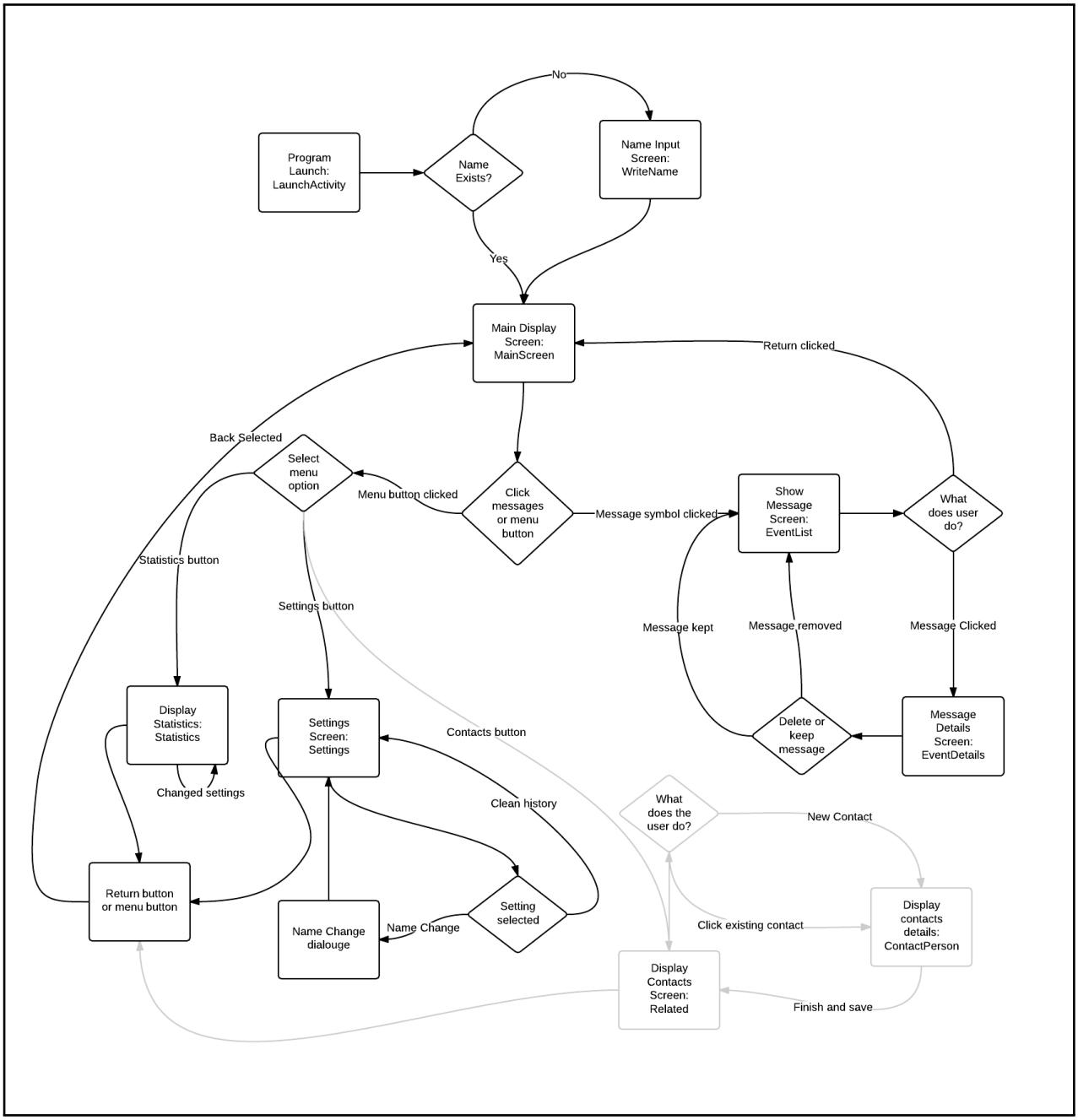


Figure 6.2: Flow chart describing the Graphical User Interface, grey lines mark features not used in the current application

The definition of Activity classes are as following:

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.

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

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

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" that 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.

[1] http://developer.android.com/reference/android/app/Activity.html

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 provdides the content provider with a basic GUI.

6.3.1 Data model

The data model describing the database handled by the Content Provider was mainly made to hold two types of information:

1. The time and number of steps, accomplished by storing receiving timestamps, and
2. The results and types of other tests, as results from other tests would also be a meaningful factor in some cases.

This is visualized in Figure 6.3.

6.4 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.

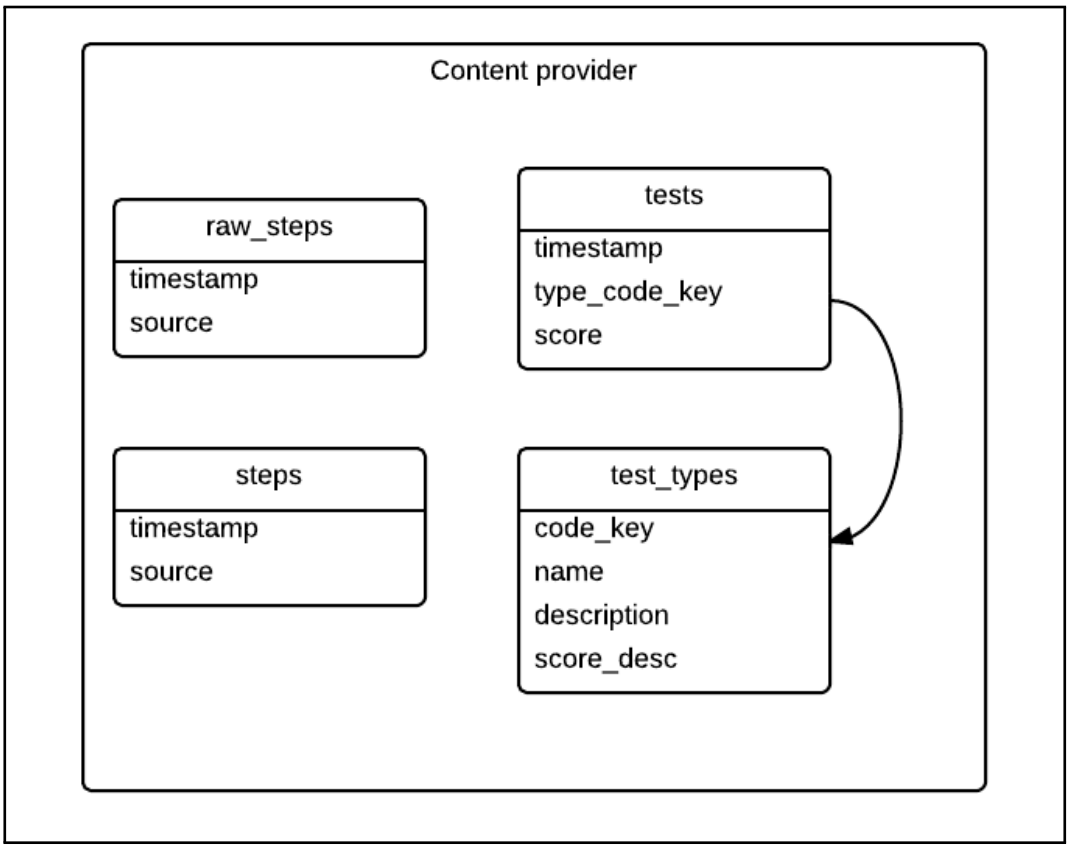
LaunchActivity handles the basic GUI for the main screen.

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

Figure 6.3: A Diagram of the Content-Providers data model



been generated[2] the StepMainService will launch a DetectStepsThread to detect steps in the given data, before clearing its data[3] 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:

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.

[2]Enough 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 in chapter 7.2), only m values will actually be used for step detection.

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

6.5 Class description and diagram

The class structures of the applications are relatively complex, as can be seen from the class diagrams in figures 6.4, **??** and 6.6. As the class diagrams can be hard to read in the paper format, it is recommended to read the digital version on the application website[4]. The class diagrams are not the recommended way to understand the structure of the code, it would be better to consult the textual description above as well as the JavaDocs[5].

6.5.1 Translations

The localization was implemented as described above, with several XML files containing the string values that was to be used in the GUI. Which language the values are appropriate for is determined by the folder name and the active language in the android device. If a folder and string value for the current language is not found by android, the default values will be used instead. This application has the default values (for all the GUI text) written in English, as that was believed to be a more widely understood language than Norwegian. If a user has the phone set to use any Norwegian language (nynorsk or bokmål), the phone will instead display all the applications text in Norwegian. To accomplish this, it was critical that all text that would show up on-screen would only be retrieved from these string files, and be stored as a code or reference where this was not possible to do directly. The database storing all the messages the user has received is the most important example of this, as it only contains a code that enables the application to determine which message should be displayed. The application will then retrieve the appropriate string, while the android operation system will decide which language file to retrieve the string from. This means that process to translate the application was quite simple, as all that was needed was to copy a few files into a new folder with the appropriate name, and rewrote the content of the string to something appropriate for the new language. Any developers who would like to alter the application to display in a different language,

[4] `http://valens.brennhe.it/#documentation`
[5] `http://valens.brennhe.it/javadoc/`

Figure 6.4: Class diagram for the demo application

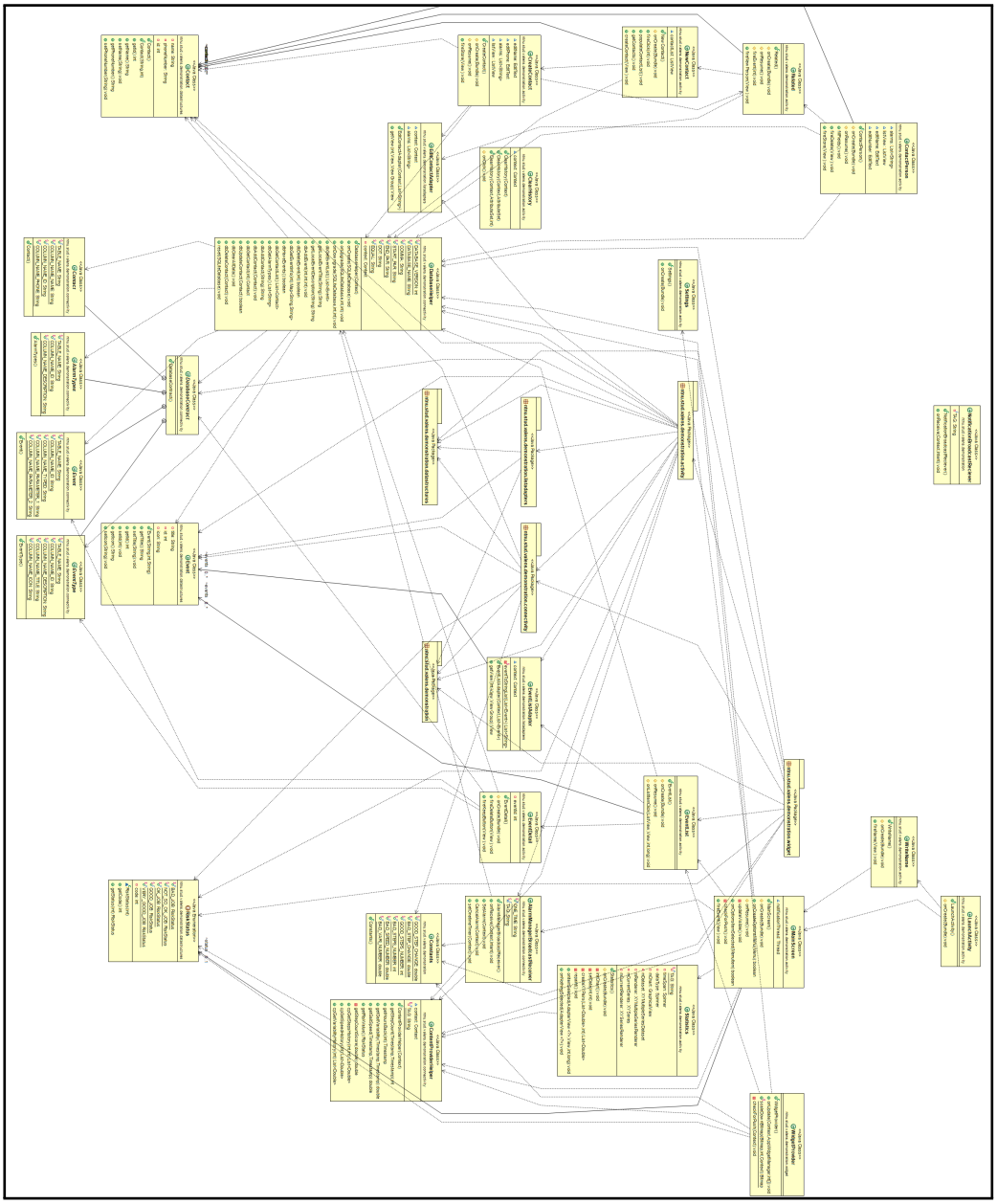


Figure 6.5: Class diagram of the content provider application

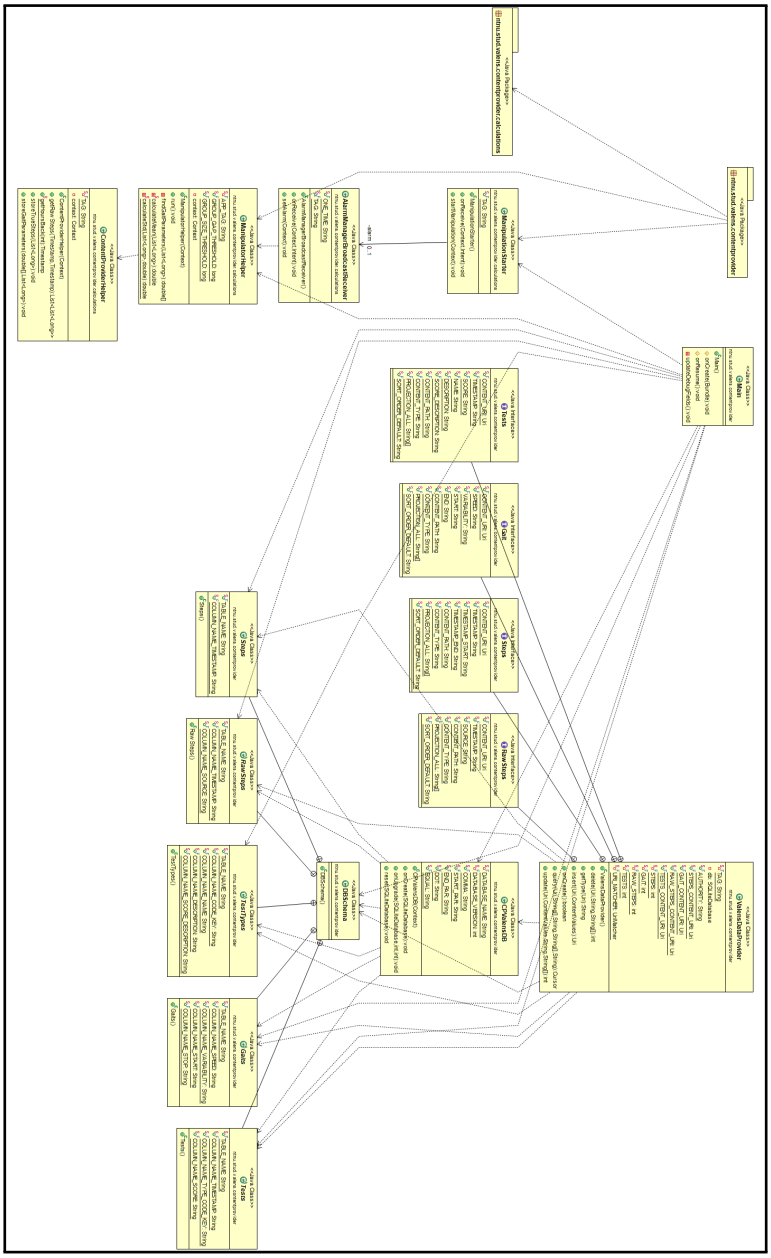
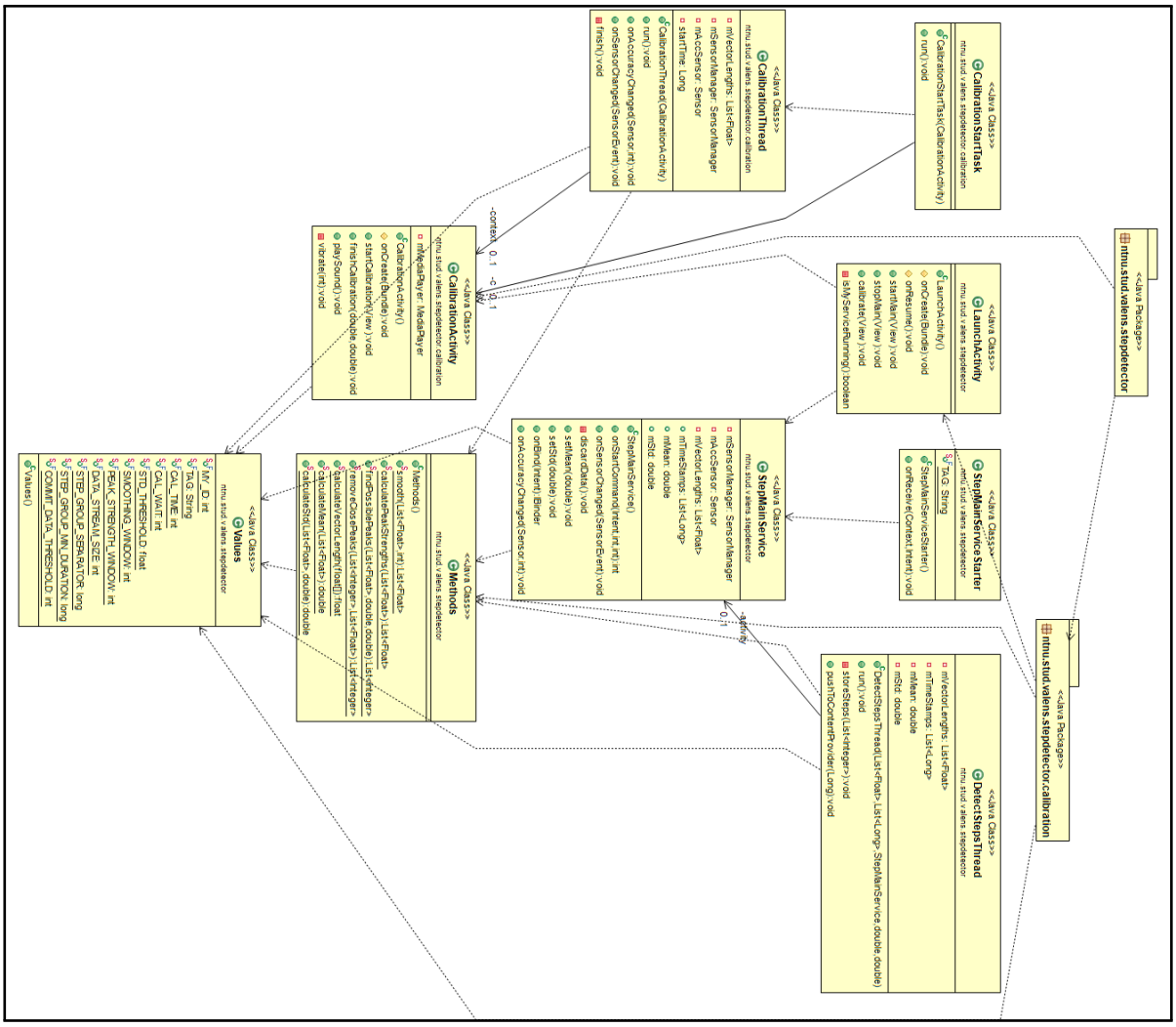


Figure 6.6: Class diagram for the step detector application



would only have to perform the same process.
The application is localized for Norwegian and English, although all that is needed for localization an knowledge of English, and the language the strings are to be translated into.

Chapter 7

Step Detection Algorithm

The algorithm used to detect steps, and its development, is described in this chapter.

7.1 Development and purpose of Algorithm

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 computer-based prototype was developed. Finally, the algorithm was implemented in the Valens Step Detector app, modified to work with live data.

7.2 Core Algorithm

The current section will explain the algorithm used for step detection, formulated as an off-line algorithm. Section 7.3 will explain what changes were made to make the algorithm work for live data streams.

Experiments showed that the accelerometer consistently gave the most

predicative feedback patterns for step detection[1]. Furthermore, it was found that the total acceleration was a better predicator 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 average[2]. 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 is an adaptation of the generic peak detection algorithm proposed by Palshikar[9]. The peak detection algorithm 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

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

where k is a constant referred to as the *peak strength window*.[3]

[1]None 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.

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

[3]This is only one of several peak strength functions proposed by Palshikar[9]. Palshikar

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 fulfills 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 [9] .

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

7.3 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.

showed that other peak strength function performed better on the test data, but this peak strength function was chosen for the current project because it is intuitive and easy to implement.

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 the 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 is non-linear and therefore 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 viable solution is therefore 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.

7.4 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 generated[4]. 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.

7.5 Gait calculations

To calculate the Gait variables, the steps detected are used as base data. The step detection data is based on time in milliseconds since 00:00:00 UTC on 1 January 1970 (Unix epoch), usually referred to as a timestamp. This makes it easy to calculate mean times, intervals and standard deviation, which is the numbers we want for the two Gait parameters. To lower the fault rate in this data, periods of walking less than 10 seconds is discarded. This because of input from health experts saying that periods less than 10 seconds usually refers to inside walking, and is not suitable to do calculations with due to differences in indoor and outdoor walking Gait parameters.
The data is calculated by a calculation service inside the Content Provider and is scheduled to run once a day. The longer the time and the larger the amount of data, the more accurate calculations is able to be done, since the calculations is based on averages.

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

7.6 Room for improvement

The current step detector is rather primitive, and has room for improvements in several areas. Testing has however shown the step detector to perform within acceptable limits, at least for a prototype-stage application. The effort required to produce significant improvements to the step detector was therefore not justifiable given the time limits of the project. However, step detector improvement appears to be a major concern if the application is to be developed further.

The following areas of improvement have been identified:

1. Because the phone is kept in a pocket, it is closer to one of the legs. This results in accelerometer output for movement in one foot being significantly higher than the other. Figure 7.1 illustrates this phenomenon. A step detection algorithm that exploits this fact will necessarily be more precise than the current algorithm, which does not. However, it is hard to see exactly such how an algorithm can be devised.

2. Several different functions can be used for peak detection, and experimentation with different functions could possibly improve detector performance.

3. Constant values could be tweaked further (see section 7.4).

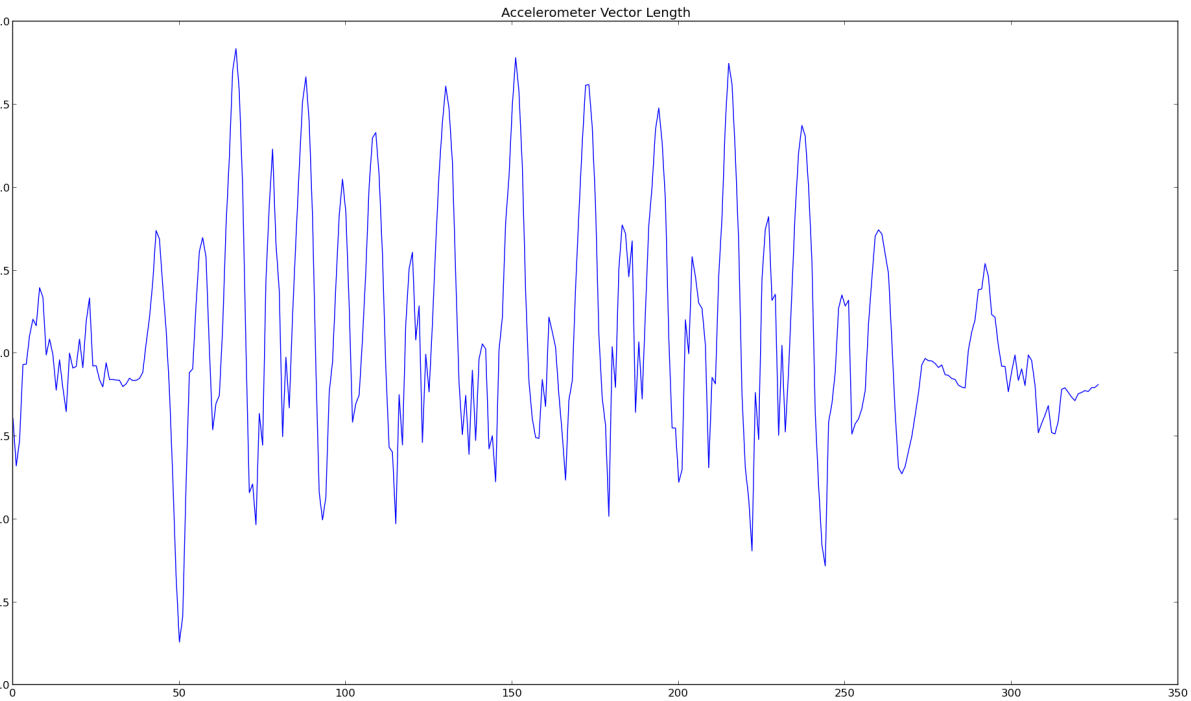


Figure 7.1: A normal pattern of walking: Notice the alternating peak height in the middle.

Chapter 8

Testing

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

8.1 Levels, methods and types

Software testing is generally broken down into several levels, methods and types. This section will introduce general testing terminology.

8.1.1 Levels

Software testing has four levels that correspond to how far into the software development process the team has come. The first three test levels refer 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. Inspections refers to a peer review or code review process where a team of individuals read through the source code to reveal early and obvious bugs. An error or bug that is not very visible for the developer may be very visible to an inspector.

3. Integration testing is the next step which takes groups of individual components to verify that they work together or to reveal faults in the integration.

4. 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.

5. 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 Section 8.1.2) to see if the customer will receive the desired and expected results.

8.1.2 Methods

The approach to software testing is 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, behavior 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.

8.1.3 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.

8.2 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.

8.2.1 Unit testing

Unit testing in the project was never done properly, nor properly documented. The method used was a form that adopted the essence of unit testing, and was done on-the-fly to verify that input correlated with output, and then removed. Most of the time it was done in-line and was not suited for reuse. Instead of heavy unit testing, it was partially replaced by Inspections.

8.2.2 Inspections

Inspections done in this project falls under the category "Code review" where part of the team or individuals read blocks of code to ensure its quality and to find bugs. A single developer working on a block of code can often get "blind", saying that they do not see obvious mistakes. Code review by a team member can speed up the process of identifying these bugs and defects for the developer to fix. This method became one of the most important types of testing to fry bugs in this project.

8.2.3 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 done for every application separately. 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 include 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).

8.2.4 System testing

System testing is the second to last level of testing, to verify that the system works together as a whole. In the project the system testing was done parallel to the last past of integration testing because of two reasons; time and complexity.

Since the app system consists of very few components the team decided to grasp the opportunity so save time and combining a few of the integration tests and system tests since they in essence overlapped each other.

The tests done in this stage is all done in the visible part of software, or also known as black box testing where you are blind to the code behind. Differing from previous methods, this part of testing had tests developed and laid out beforehand and all the faults, bugs and defects were noted according to these tests and afterwards fixed by a developer before commencing new tests. This because an error or bug may be connected and one bug can cause

or solve another.

8.2.5 Acceptance testing

The last level before handing over the final application to the customer, but tested 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.

8.2.6 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 too much for the target group.

8.2.7 Acceptance testing and deployment

During the end of the project, the customer expressed a desire to do acceptance testing on the application. The application was deployed on a web site [1] along with instructions for installation and a form to give feedback. There was not much feedback, but the customer did send a mail with feedback, which is included in Appendix C.

8.3 Tests used under development

Here goes the tests for the GUI and demo application:

8.3.1 GUI Tests

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

[1] `http://valens.brennhe.it/`

Info	
Title	Activity start
TestID	01
Writer	Everyone
Developer	Not available
Date created	15.02.13
Date tested	15.02.13
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	Everyone
Date created	22.03.13
Date tested	01.03.13
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 9

Conclusion

In this chapter there is information concluding the project, such as what the group learned and experienced.

9.1 Lessons learned

9.1.1 New Experiences

All members of the group learned much about teamwork, project management, planning, programming in a group, programming towards the Android platform, programming documents in LaTeX, and writing documentation for project activities.

9.1.2 Learning experiences from development and tool use

Following are things that relate to project management and software development that the group learned during the project:

- It is important to ensure that documentation is done in a systematic fashion. In the beginning, a large amount of the reports and documentation that the group made for internal use, turned out to create more work and confusion later on. This was in part because it was

not thought to be necessary at the time. Looking back, it seems like it would have been much more efficient to write reports well, and write them in LaTeX while they were still new.

- Communication and what tools to use, was in flux for some time. Even when a few tools were settled on, the group still had problems with making sure all the members were on the same page regarding what needed to be done.

- Usage of tools which not all members were equally experienced with turned out to be problematic. This was because of misunderstanding and difficulties which could often be resolved by only one or two members, and slowed down the entire group. This might have been mitigated if all the members had a better understanding of the tools that were used. This could be accomplished by teaching sessions and group members being encouraged to seek a better understanding on their own.

9.1.3 Lessons learned about Research and Prestudies

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 was 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.

9.2 Conflict handling

There were no one-on-one conflicts, but some conflicts arose right after midterm regarding work load and willingness to prioritize the project. To try to solve this the group leader communicated the issues and its potential solutions via e-mail, with no further response. Some group members took the e-mail seriously and stepped up their work load, but on the other hand, some members ignored this wake up call completely and did not take any action. A meeting with the supervisor was scheduled to try to get an idea to solve the situation, but was used for other purposes instead, because the group did not have an opportunity to discuss matters beforehand. Due to the lack of a proper meeting, the situation did not get resolved. Some of the team members still prioritized otherwise, while others worked more. Technical disagreements were solved by discussing the possible solutions with the entire group, until all the group members agreed on a course of action.

9.3 Further work

Even though the project is finished, there will always be improvements to make, bugs to fix, and features to add. Some of them are listed below:

Fix bugs discovered during acceptance testing.

Research and improve the Step Detector according to section 7.6 on Room for improvements.

Extend the Step Detector to measure stride length with GPS.

Extend the Health Helper with the ability to do personal tests and measure inactivity elaborated in section 2.3.

Improve the Health Helper by adding features suggested during acceptance testing.

Research critical thresholds for fall risk. The thresholds currently used are not based on empirical data.

Research better messages to display to users of the Health Helper. The messages shown now are too similar and makes little to none sense.

Proper testing throughout the system. The acceptance testing should be done by more than one person with one device to ensure compatibility with several devices.

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] "Preventing Falls: How to Develop Community-based Fall Prevention Programs for Older Adults", Centers for Disease Control and Prevention.

[3] "Muscle strength, activity, housing and the risk of falls in elderly people." Wickham C et al., Age and ageing, 1989, 18:47-51

[4] "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.

[5] "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.

[6] "What you can do to prevent falls", Centers for Disease Control and Prevention.

[7] "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.

[8] "Software Engineering" 2011, Ian Sommerville, Pearson Education, Boston.

[9] "Simple Algorithms for Peak Detection in Time-Series", Girish Keshav Palshikar, Tata Research Development and Design Centre (TRDDC).

Appendix A

WBS

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, A.2, A.3, A.4, A.5, A.6, and A.7.

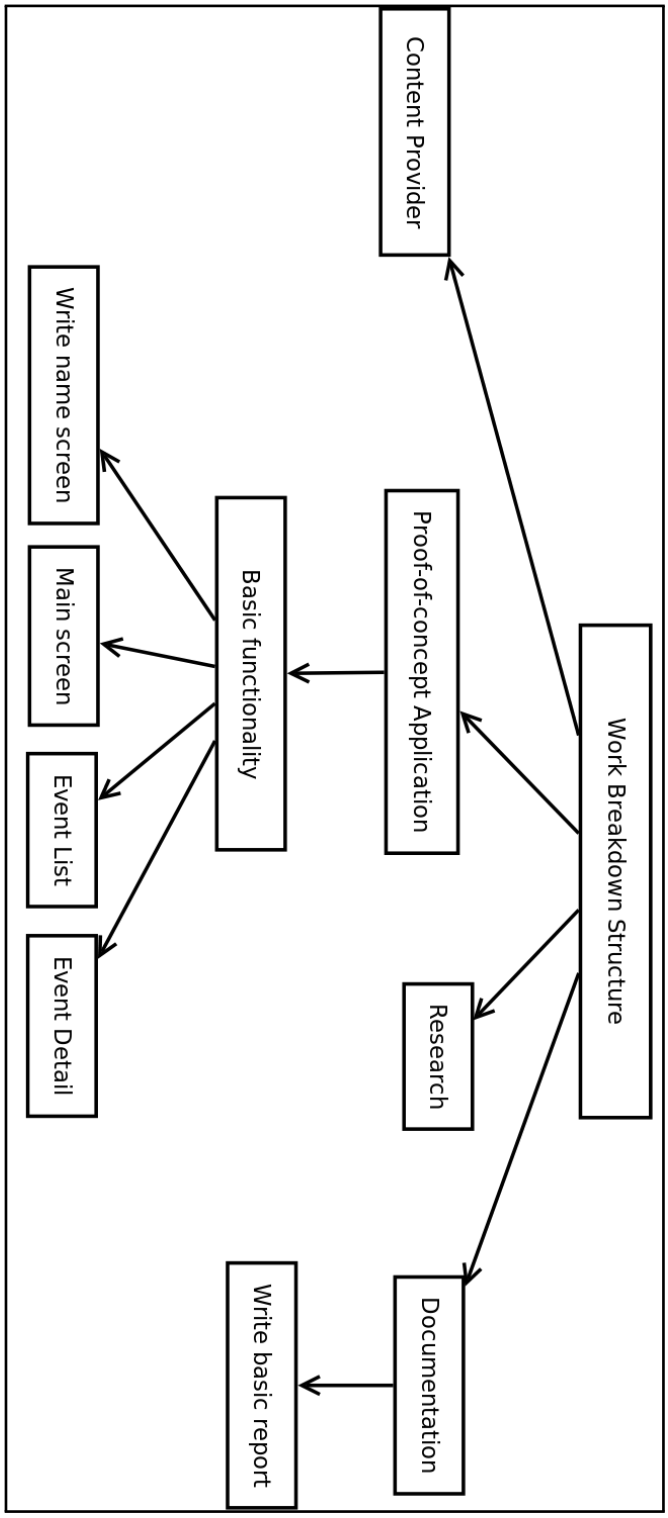


Figure A.1: The WBS as per 22.2.13

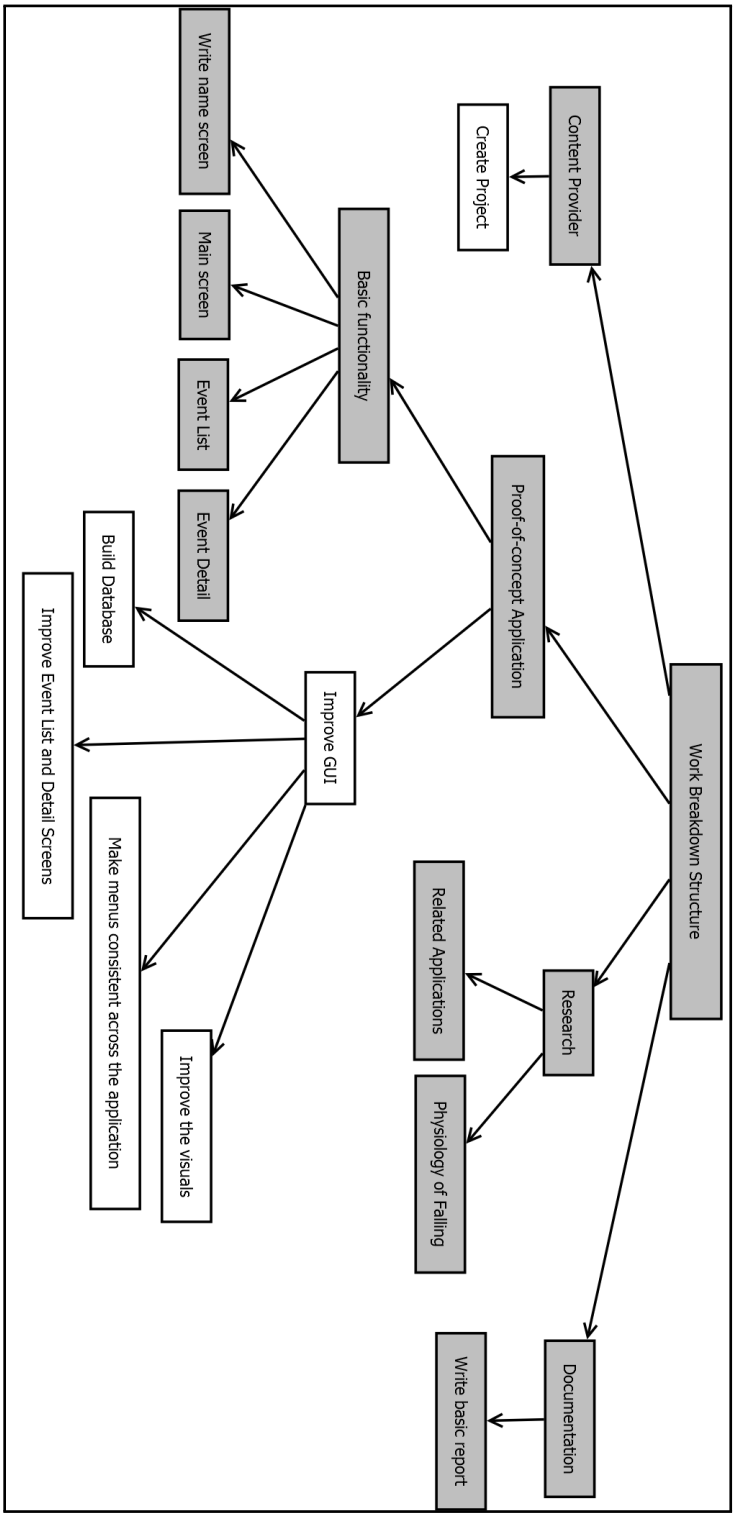


Figure A.2: The WBS as per 1.3.13

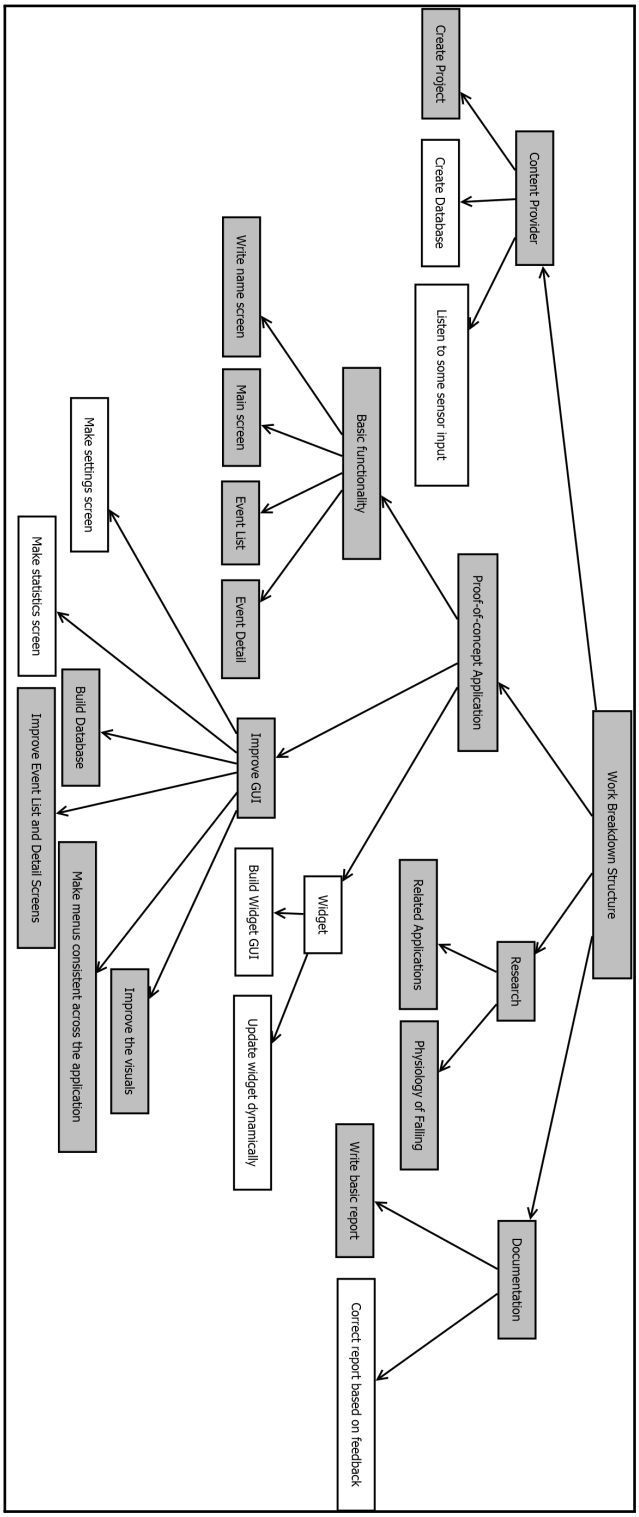


Figure A.3: The WBS as per 8.3.13

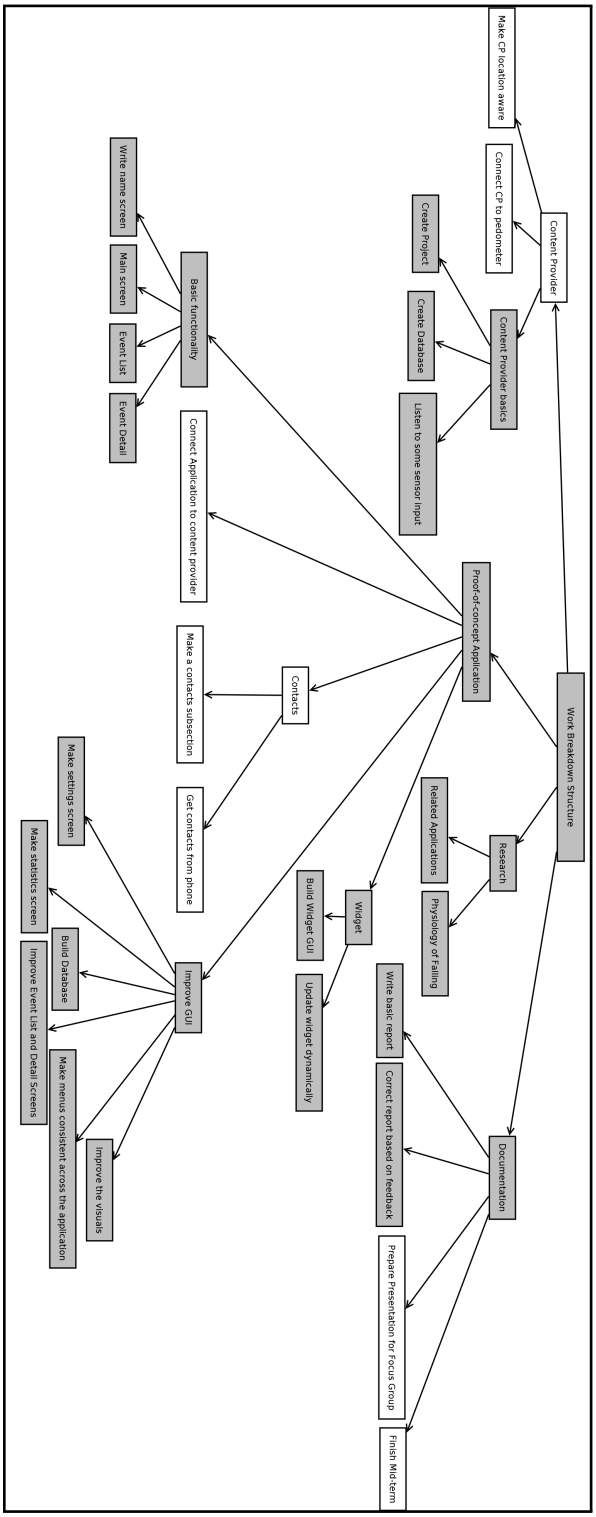


Figure A.4: The WBS as per 15.3.13

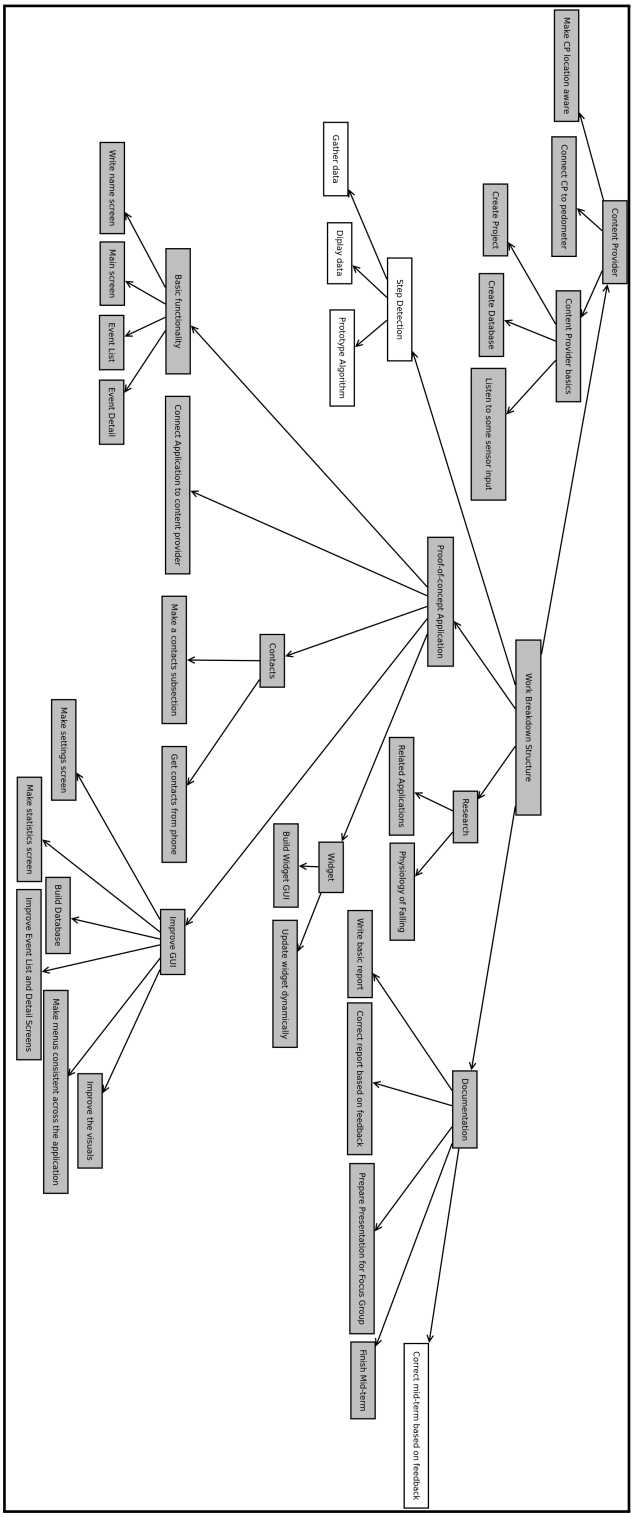


Figure A.5: The WBS as per 22.3.13

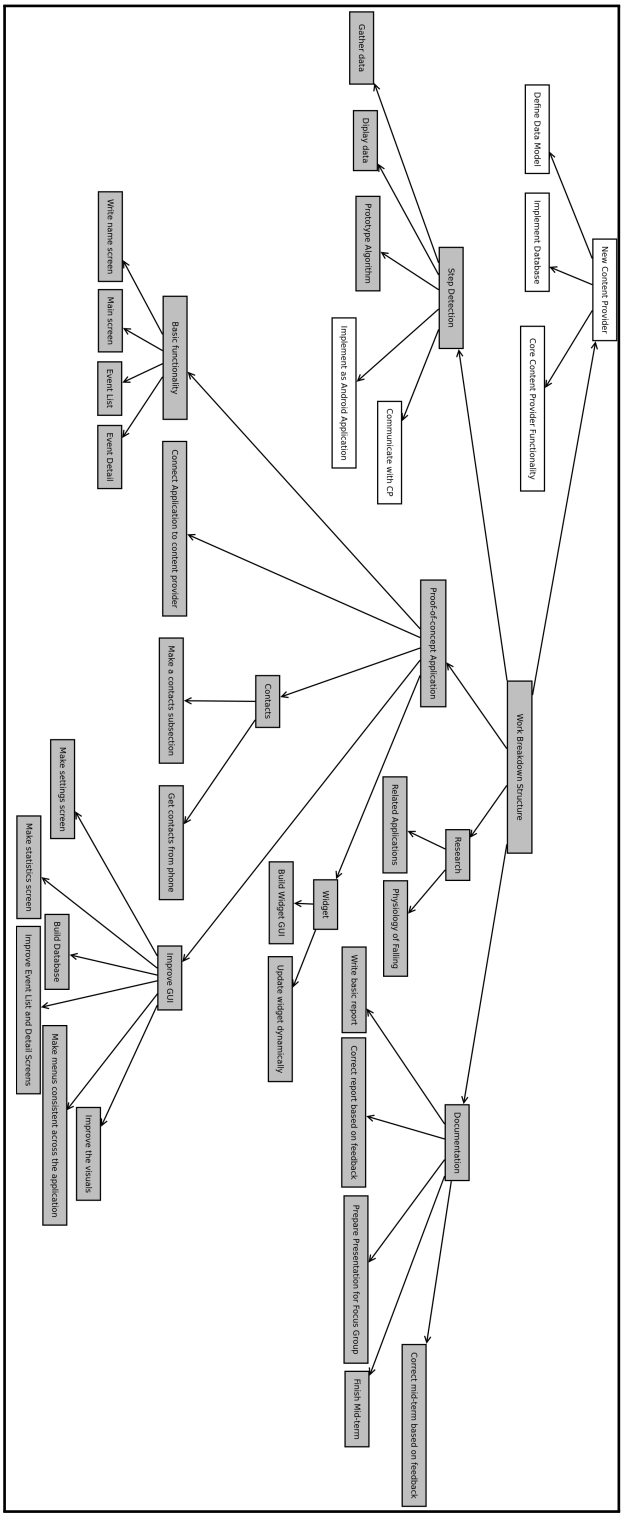


Figure A.6: The WBS as per 5.4.13

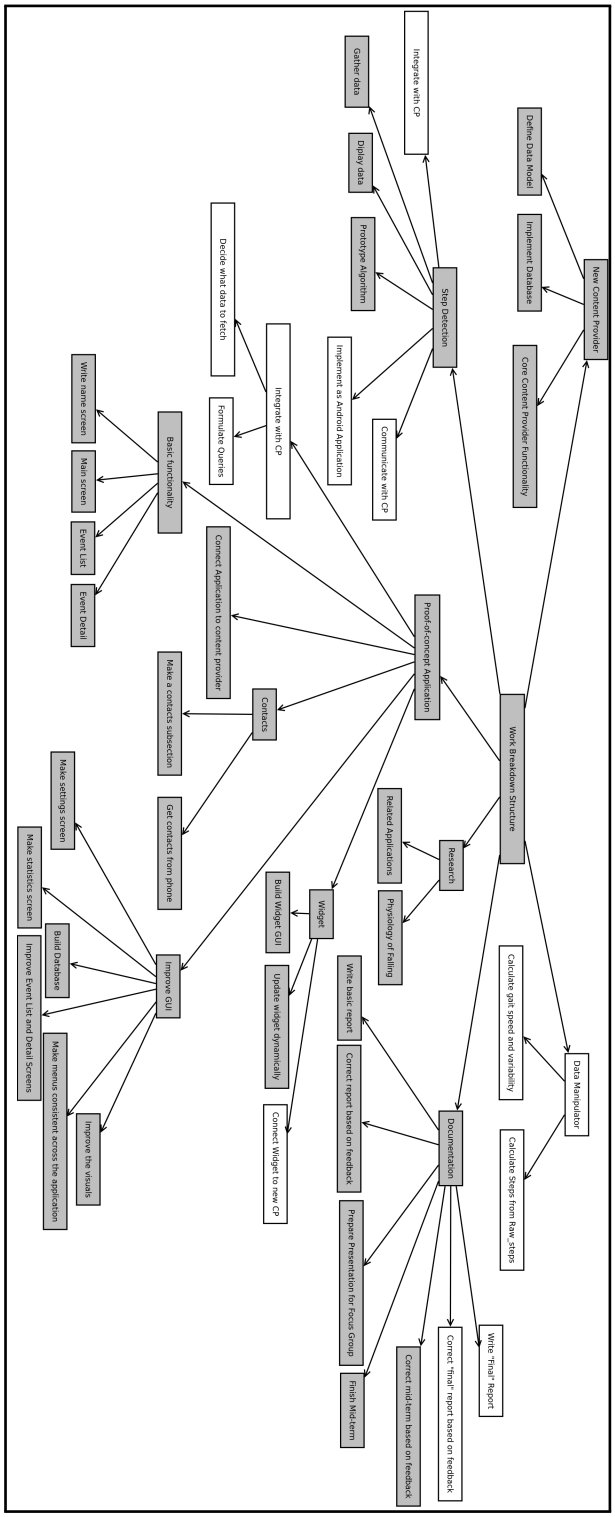


Figure A.7: The final WBS

Appendix B

Work summary

B.1 Sprint summaries

The following is a list of the sprints:

Sprint 1: 01.02.13 - 08.02.13

User stories and paper prototypes of the GUI were developed.

Sprint 2: 08.02.13 - 15.02.13

The group focused on learning Android development. New GUI paper prototypes were developed based on customer feedback. A mock-up application demonstrating core parts of the GUI was developed.

Sprint 3: 15.02.13 - 22.02.13

Continued efforts on learning Android development. The mock-up was developed further. Research was conducted on the application domain, and on related applications.

Sprint 4: 22.02.13 - 01.03.13

The prototype application was expanded with a database, GUI visuals were improved, menu usage was made consistent throughout the prototype. The group tried to familiarize themselves with Content Providers in Android.

Sprint 5: 01.03.13 - 08.03.13

A Content Provider was developed, incorporated the step detection from the Pedometer open-source application. Some secondary screens (settings and statistics) were implemented. Basic widget created.

Sprint 6: 08.03.13 - 15.03.13

Widget was connected successfully to the main application. Application connected to the Content Provider. Work on report for the mid-term hand-in. Presentation for the focus group prepared. Unsuccessful

attempts at making the Pedometer/Content Provider location aware. Implementation of contact management.

Sprint 7: 15.03.13 - 22.03.13

Research and development of step detection algorithm: Sensor data collection application developed, python scripts for displaying data and prototyping of algorithm. Work on Javadoc and code re-factoring. Correction of report based on mid-term feedback.

Ester holidays: 23.03.13 - 02.04.13

Little work was done due to the holidays.

Sprint 9: 03.04.13 - 05.04.13

New system architecture was designed. Step detection algorithm implemented as Android application. The old Content Provider/Pedometer discarded, and new Content Provider implemented.

Sprint 10: 05.04.13 - 12.04.13

Efforts were focused on bug-fixing and integration of the three components.

Sprint 11: 13.04.13 - 19.04.13

Work on final report and small adjustments the integration of the components. The code underwent re-factoring and commenting.

String 12:20.04.13 - 10.5.13

implemented alarmmanager-timing for updating, sending notifications and doing calculations, localization of db content, recreated statistics screen, deployed web page for installation and feedback, simplified for-

mula for user feedback.

B.2 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 avaliable, 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 that 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.

Appendix C

Test results

Here is a copy of feedback the group received from the customer when the application was deployed for testing.

Web page

Babak Farshchian <Babak.Farshchian@sintef.no> 12:45 23. mai 2013
Til: Filip Andre Larsen Tomren <filiptomren@gmail.com>
Kopi: Johannes Vassdal <johannes.vassdal@gmail.com>, "datdanny@hotmail.com" <datdanny@hotmail.com>, "eliasaa@stud.ntnu.no" <eliasaa@stud.ntnu.no>, Tayfun Kücükyareli <tayfun.kucukyareli@live.de>

Hi unfortunately I have been the only tester of the application. I write my test report here. See whether you can include it in your report. The reason for not having anyone else is that we have had a couple of project meetings in May. Bad planning, not because people did not like your app!

Settings:

- I have Android 4.1 on HTC One+
- I installed your apps two weeks ago and have had them on my phone with measuring on.
- I have tried to carry my phone in my pocket for pedometer to work correctly.

Positive things:

- The app was very easy for me to set up. I had some problems with the calibration process and I reported that you could add info about when successful calibration was done last time. Otherwise very intuitive setup.
- The application has been extremely stable. I have not had a single crash. I also use the widget. Widgets normally cause crash but yours has not done so.
- Battery usage: The app uses around 2-3 % of total battery consumption. It is very good and is one of the apps that uses least power even if it is always on.
- The widget is nice and works as it should.

Not so positive things:

- I think there is a bug when counting steps. I get messages every day, and they use different numbers for each day. Example: in one message I have walked 1089 steps on May 21, in another message I have waked 375 steps.
- Messages are very monotonous. I only get one type of message "You need to move more". I wonder if this has to do with the bug above.
- Widget and messages are not consistent with each other. Even when widget shows a green smiley I still get messages that say "You need to move more" and the numbers always show a reduction in number of steps even with a green widget.
- It is difficult to use the statistics part. Steps are more or less ok, even if the fonts are very small all over the statistics page. The other two variables on gait I think show meaningless data. It is at least not possible for me to draw a proper conclusion.

Suggestions for improvements:

- The above "bugs" are annoying and make the application much less useful. Good if they can be fixed or if you can describe in your report how they can be fixed.

- Notifications should be marked read after the user reads them. Or you should have an "archive" button in addition to keep and delete.

- When you click on widget and go into message list it should be possible to access the application menu. As it is now the widget is not very useful as it only shows a list of messages when clicked on. If you want to go to the menu you have to exist from the widget and click on the application icon instead.

-

/babak

Voice +47 992 86 869

C.1 Summary of interview with medical professionals

The group had a interview with medical professionals, and got an improved understanding of the problems they faced, and parts of the application that needed alterations.

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.

The requirements for the interface was altered by an informal usability test done with the medical experts. Their feedback is summarized here:

- 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.

Appendix D

Time-sheet

Here is the time-sheet used to log time spent working on the project.

Time sheet

	Total:	27.01	28.01	29.01	30.01	31.01	01.02	02.02	03.02	04.02	05.02	06.02	07.02	08.02	09.02	10.02	11.02	12.02	13.02	14.02	15.02	16.02	17.02
Tayfun	22	4	1	2		2			1	3	2	2	1		2						2		
Dat	23	4	1						1	5		4		3				3	2				
Elias	32	4	1	2					1	5		2	1	3				7	2		3		1
Filip	25	4	0				1		1			4		3	2	1		6			3	*[1]*	*[1]*
Johannes	30	4	1				1		1	5		4		3	1			3	2	2	3		
		Week 5								**Week 6**							**Week 7**						
						Total			37				Total			56				Total			39
		Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun
	Total:		18.02	19.02	20.02	21.02	22.02	23.02	24.02	25.02	26.02	27.02	28.02	01.03	02.03	03.03	04.03	05.03	06.03	07.03	08.03	09.03	10.03
Tayfun	23					2	2					6			2			2	6	3			
Dat	22				2		2		1	2		6	2					1	4		2		
Elias	56		7	7	4		3			4	4	4		6			4	3	6		4		
Filip	38		*[1]*	*[1]*	*[1]*	2	3	4	2		5	6	2					2	6	3	3		
Johannes	37			4	2	2					2	6	2	4				3	4	4	2		2
			Week 8								**Week 9**							**Week 10**					
						Total			49				Total			63				Total			63,5
			Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun
	Total:		11.03	12.03	13.03	14.03	15.03	16.03	17.03	18.03	19.03	20.03	21.03	22.03	23.03	24.03	25.03	26.03	27.03	28.03	29.03	30.03	31.03
Tayfun	35			2	3	3	8		2	1	1	5		1		2			3	2	2		
Dat	22		1		5		7					3		4			2						
Elias	27			2	5		7				8	2		3									
Filip	31			[2]	2[2]	11	8				4	6											
Johannes	25			1	5		8				6	5											
			Week 11								**Week 12**							**Week 13**					
						Total			80				Total			51				Total			9
			Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun
																	Easter vacation						
	Total:		01.04	02.04	03.04	04.04	05.04	06.04	07.04	08.04	09.04	10.04	11.04	12.04	13.04	14.04	15.04	16.04	17.04	18.04	19.04	20.04	21.04
Tayfun	20				1	4			2			2	3	3		2					3		
Dat	28				1		3			2	3	3	1	5				3	4	3			
Elias	44				3		3			2	5	5		4			3	5	5		9		
Filip	38				3		2		6		4			2				5	7	1	8		
Johannes	55						3			3	3	3	2	5	2	1	6	5	7	4	11		
			Week 14								**Week 15**							**Week 16**					
			Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun
		Easter vacation																					
	Total:		22.04	23.04	24.04	25.04	26.04	27.04	28.04	29.04	30.04	01.05	02.05	03.05	04.05	05.05	06.05	07.05	08.05	09.05	10.05	11.05	12.05
Tayfun	11			1	5					1	4												
Dat	41				7	2				4	6	6	4	4				4		4			
Elias	45			2	5		6			4	6	6	4	7			1	4					
Filip	31				4		6	3			6			8				4					
Johannes	39				7	2				2	6	6	4		2		1	5		4			
			Week 17								**Week 18**							**Week 19**					
			Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun
																				Exam period			
	Total:		13.05	14.05	15.05	16.05	17.05	18.05	19.05	20.05	21.05	22.05	23.05	24.05	25.05	26.05	27.05	28.05	29.05	30.05	31.05	01.06	02.06
Tayfun	1																1						
Dat	19										3	3	3				10						
Elias	28			2												11	15						
Filip	31								2	2	1				2	9	15						
Johannes	22			2												5	15						
			Week 20								**Week 21**							**Week 22**					
			Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun	Man	Tue	Wed	Thur	Fri	Sat	Sun
			Exam period																				

	Overall total:
Tayfun	112
Dat	155
Elias	232
Filip	194
Johannes	208

[1] Vacation - could not work on project

[2] Sickness - major headache rendered me useless