May 27

Final Report

The Team Naegling

Team Members:
Johan Nilsson
David M. Szabo
Elsa Wide
Henrik Edholm
Simonas Stirbys
Mikaela Lidström

Supervisor: William Granli

Table of Contents

1. Overview

1.1 Introduction

In this section of the Software documentation you shall obtain information about the purpose of the developed Software System, including the scope, description of the system.

1.2 Purpose

The purpose of this document is to present a detailed description of the City of Assassins System. It will explain the features and the purpose of the system, the interfaces, what the system will do, the constraints under which it must operate and how it will react to external factors. This document is intended for both the stakeholders and the developers of the entity.

1.3 The scope of the document

The scope of the document is to show the different part of the system, how they interact with each other, how the work was organized and what risks affected our work.

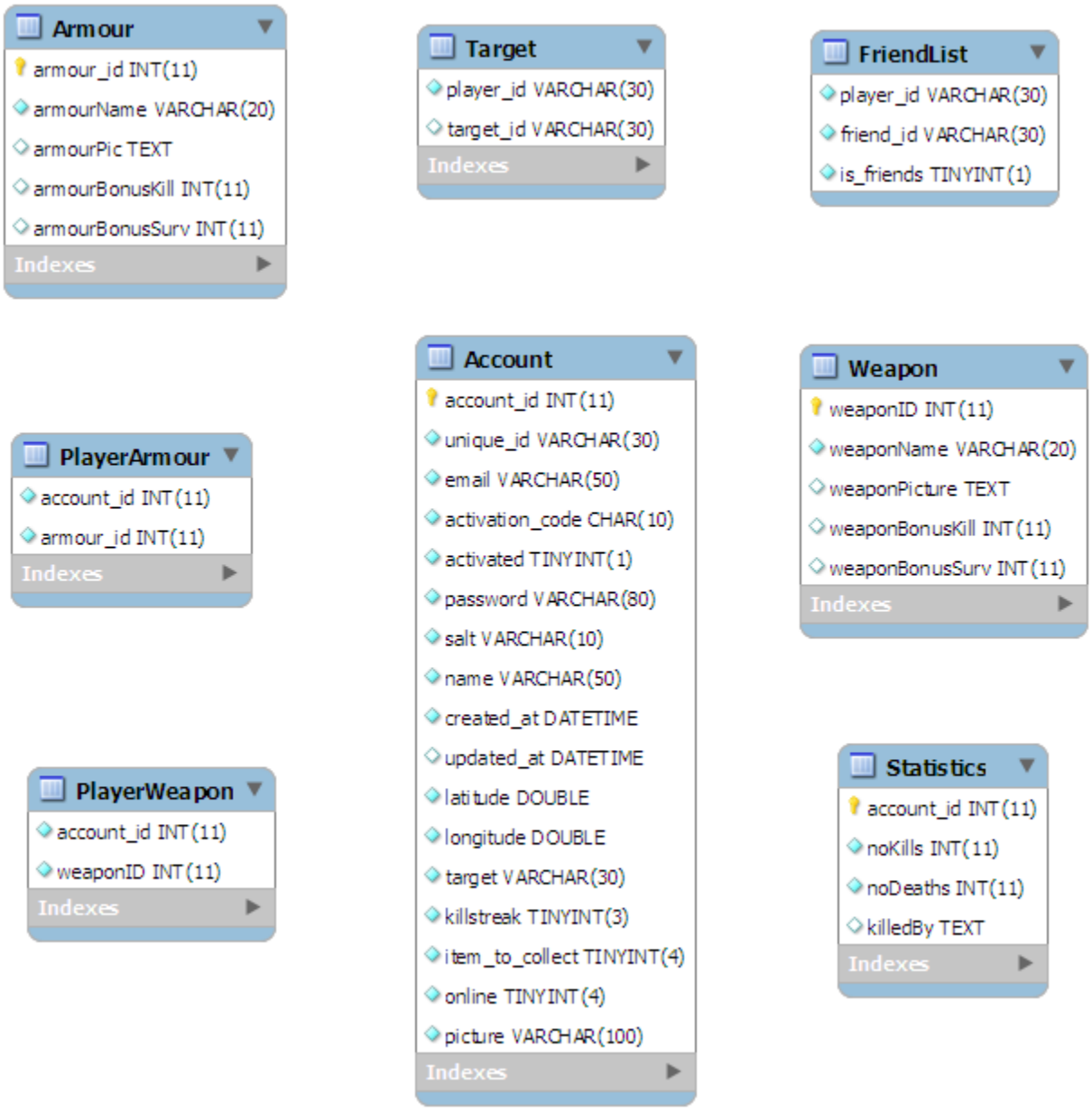
1.4 Overview of the document

The Final Report is divided into 5 sections. The first is the introduction. The second is about the use cases, to understand how the system works. The third one is the work distribution. The fourth one is about the risks. The fifth one is about the changes.

1.5 General System description

The application is a GPS based game that takes the use of a Raspberry Pi to gain more functions. The users of this application are able to have a profile, add friends, assassinate targets within a 1.5 mil radius. They can acquire items to increase survivability and assassination odds, change their profile picture as well as acquiring and watching rankings. The system includes an android application, a webserver and a raspberry pi. In the following section it will be explained what each system was doing.

1.6 Webserver



Created by David M. Szabo, 2014.05.26, Database diagram.

The webserver includes a database that communicates with the android application and the raspberry pi.

1.7 Android application

The android application interacts with the database and the raspberry pi. It has a register, log in, a main activity, where the players can play, a Change Password, Forgot Password, NFC,

Profile and a Ranking Activity. The main activity shows a Google Maps and includes all of the functions, which will be described later in the use cases section.

1.8 Raspberry Pi

The user can interact with the raspberry pi through the app with the use of near-field communication. With the use of this near-field communication technology the user are able to both take new profile pictures along with collecting items.

1.9 Definition and acronyms

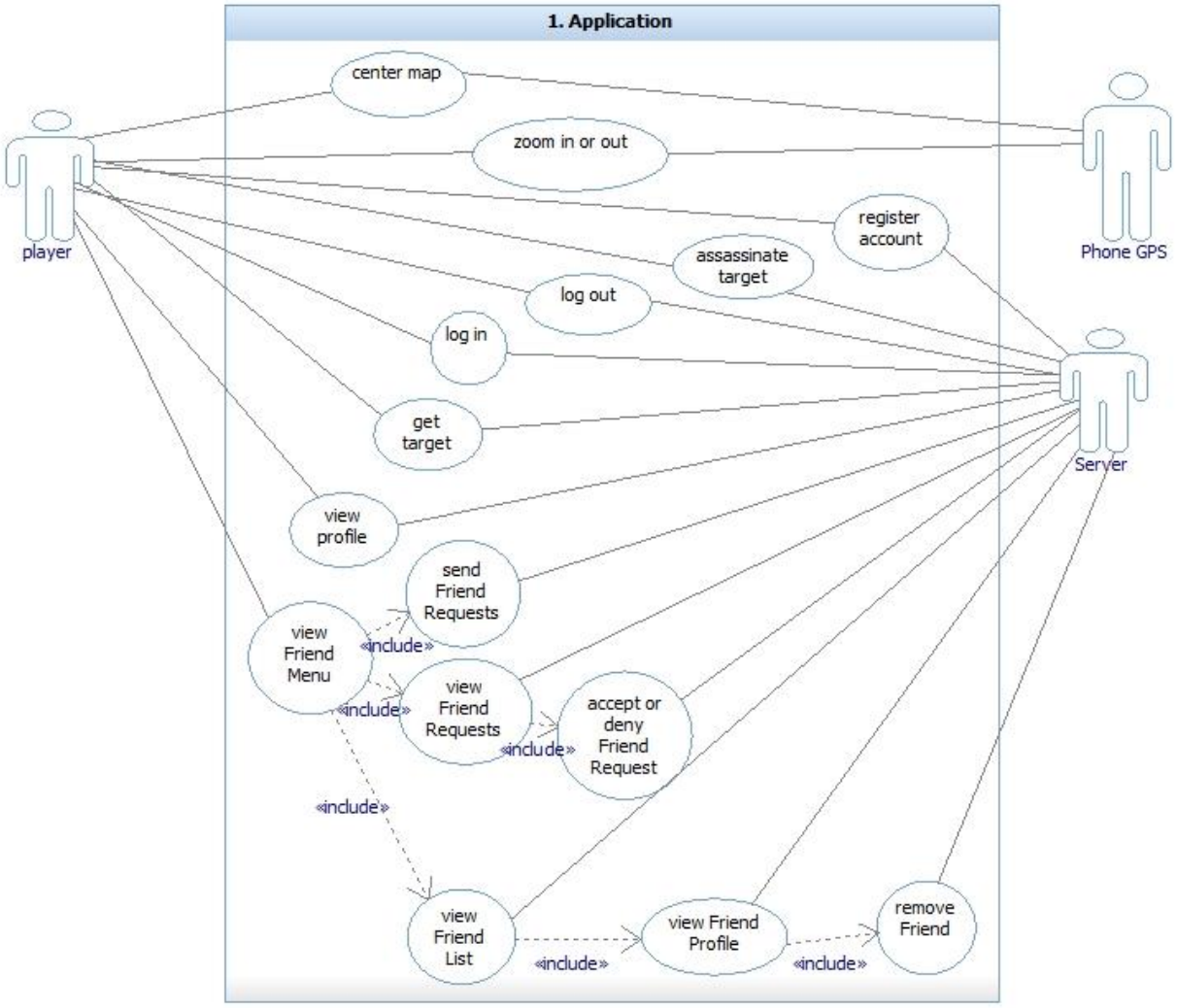
Github	"It is a web-based hosting service for software development projects that use the Git revision control system".[1]
PHP	It "is a server-side scripting language designed for web development but also used as a general-purpose programming language"[2].
Webserver	Mainly webservers are used to host websites.
Eclipse	It "is an integrated development environment" (IDE).
IDE	It "is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs offer Intelligent code completion features"[3].
Pivotaltracker	"Tracker is a simple, story-based project planning tool that allows teams to collaborate and react instantly to real-world changes. It's based on agile software development methods, but it can be used on a variety of types of projects. Tracker frees you up to focus on getting things done, without getting bogged down keeping your plans in sync with reality"[4].
Google Hangouts	It is similar to Dropbox, most commonly used to communicate with others through the internet. It is free and video chat is enabled.
Android Studio	It is an IDE for those who want to develop android applications.
Java	"It is a computer programming language that is concurrent, class-based, object-oriented, and specifically

[1] http://en.wikipedia.org/wiki/GitHub, downloaded 2014.05.20.
[2] http://en.wikipedia.org/wiki/PHP, downloaded 2014.05.20.
[3] http://en.wikipedia.org/wiki/Integrated_development_environment, downloaded 2014.05.20.
[4] https://www.pivotaltracker.com/help/gettingstarted, downloaded 2014.05.20.

	designed to have as few implementation dependencies as possible”[5].

[5] http://en.wikipedia.org/wiki/Java_(programming_language), downloaded 2014.05.20.

2. Use Cases[6]



Created by Simonas Stirbys, 20.05.2014., Hub Use Case Diagram

Application

1. Register Account

   The Player creates an account in the game.

2. Zoom in / out

   The Player can zoom in and out on the map.

3. Log out

   The Player logs out from the game.

4. Get Target

   The Player gets a new target.Display users

[6] For more informations see the Software Architecture document.

Describes how users are displayed on the map.

5. View Profile

   The Player views his / her profile.

6. Send Friend Request

   The Player sends a friend request.

7. View Friend Requests

   The Player views his / her friend requests.

8. Log in

   Log in function allows Players to log into their Player account.

9. Center Map

   Center map use case zooms to the Player's location on the map.

10. Assassinate Target

    The use case allows the Player to assassinate his / her target.

11. View Friend Menu

    View Friend menu use case allows Players to handle System options concerning their friends.

12. View Friend List

    View Friend List use case displays the Player all of his friends in the game.

13. View Friend Profile

    View Friend Profile use case allows Players to view their friend's information.

14. Remove Friend

    This use case allows Player to remove his friend from his friend list.

15. Accept or Deny Friend Request

    Accept or Deny Friend Request use case allows Players to accept or deny friend requests.

16. Forgot Password

    The Player forgot his / her password use case allows user to get a new password.

17. Change Password

    This use case allows the Player to changes his / her password.

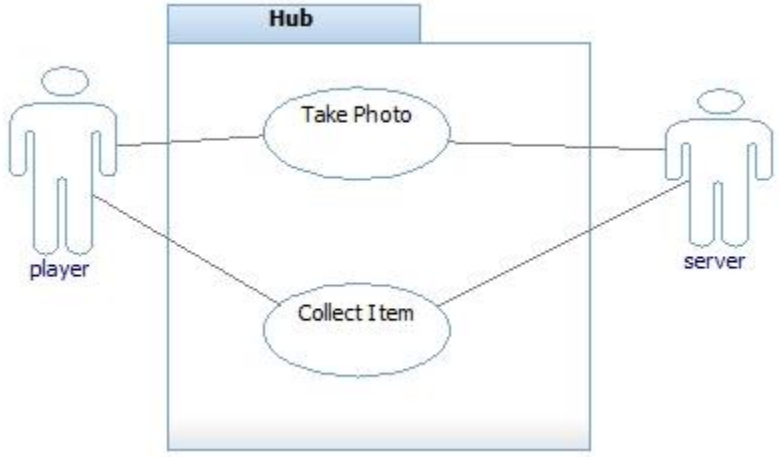
18. View Ranking

The player views the ranking of all players.

19. View Friends Ranking

The player views the ranking of all players.

HUB



Created by Simonas Stirbys, 20.05.2014., Hub Use Case Diagram

20. Take a Photo

The Player can take his/her profile picture for the application from the PI.

21. Collect Item

When the Player assassinates the target he / she can collect item from the PI.

3. Work Distribution and Group Dynamics

The Naegling group consists of: Johan Nilsson, Mikaela Lidström, Simonas Stirbys, David M. Szabo, Elsa Wide, Felix Fortoul and Henrik Edholm. Our supervisor was William Granli.

3.1 Organization and Tools

3.1.1 Work organization

The group has a social contract that characterizes the general requirements of being a part of the group. The social contract includes information about the meetings, project work, breach of contract and communication.

The work was organized through scrum. The Naegling group had daily scrum meeting every working day at 9 o'clock. The team also had Scrum review, Scrum retrospectives and Scrum planning every second Wednesday, thus 2 weeks sprints. David M. Szabo was the Scrum Master.

In the beginning the group had 5 members and this number was increased to 7, Elsa Wide and Simonas Stirbys joined us. We organized our work in the beginning by working closely together doing a lot of research and practicing Python and Android, getting to know each other and the new different elements that we had to work with. After a while we split up the team, assigning specific individual tasks to each person every sprint.

3.1.2 Group Distribution

In the beginning, in February, we divided the people in 3 groups, the hub (2 people), the android app (2 people), and the assassination group (3 people). They had to research in their areas and we defined in what they had to research. This research was about: "how to use the pi, how to do python coding, how to github, and how to use the Google Maps plugin". Everyone had to watch youtube series, tutorials and practice. The android app team had to research 8 hours – 6 hours about the android development and 2 hours about Google Maps. The hub team had to research 3 hours about the Raspberry Pi and 8 hours about python language. The github research was done by one person for 3 days and she had to educate the others later on.
Then, in March, then we renamed the groups into Python (2 people), Android (2 people) and Game design (3 people). Every group had to spend 3 days on research. We modified the definition of done sentences after talking with Imed, our product owner, and came up with these definition of dones: for the android group - make an app with Google Maps implemented and a GPS location shown, for the python group - know how to pass information from the server to the pi and vice versa, connection to database and one query, for the game design group - a basic layout of the gameplay, domain model and sequence diagram.

Afterwards, these teams started coding and their design and when the python team and the game design team were done with their part everybody joined the android team to start developing features on it. The android team educated them and assigned tasks to them for faster integration.

3.1.3 Tools

We used the following tools for the project work.

Git: The code sharing was organized through Git. Everybody had his/her own branch and we had a master branch, where the core code was. If somebody had to work on an issue, he pulled the master branch to his branch and then he was experimenting things in his own branch. When this person was ready with the issue, then he merged his code to the master branch.

Pivotaltracker: This tool was used to organize the backlog, show burn down or burn up charts and to assign tasks to different people. It has an ice box, current backlog, done log features and you can see the groups velocity by every sprint.

Google hangouts: We held the daily scrum meetings on google hangouts, as not everybody was present in school every workday of the week.

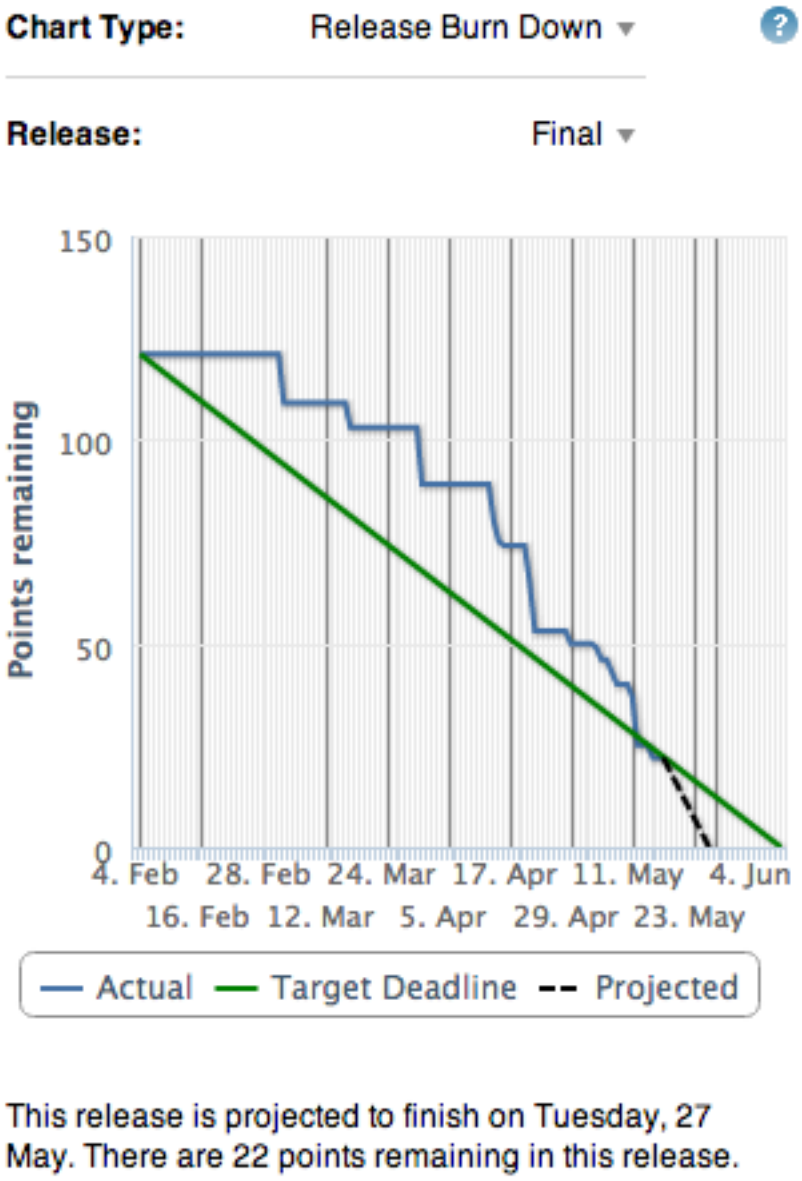
Facebook group: We used the facebook group to message each other with the latest updates about school and project.

Dropbox: This tool was used to store all the project related content aside code. We stored the design diagrams, the pictures for the game, the final documentation files, audio files.

Eclipse: We used Eclipse to write our Java-android codes.

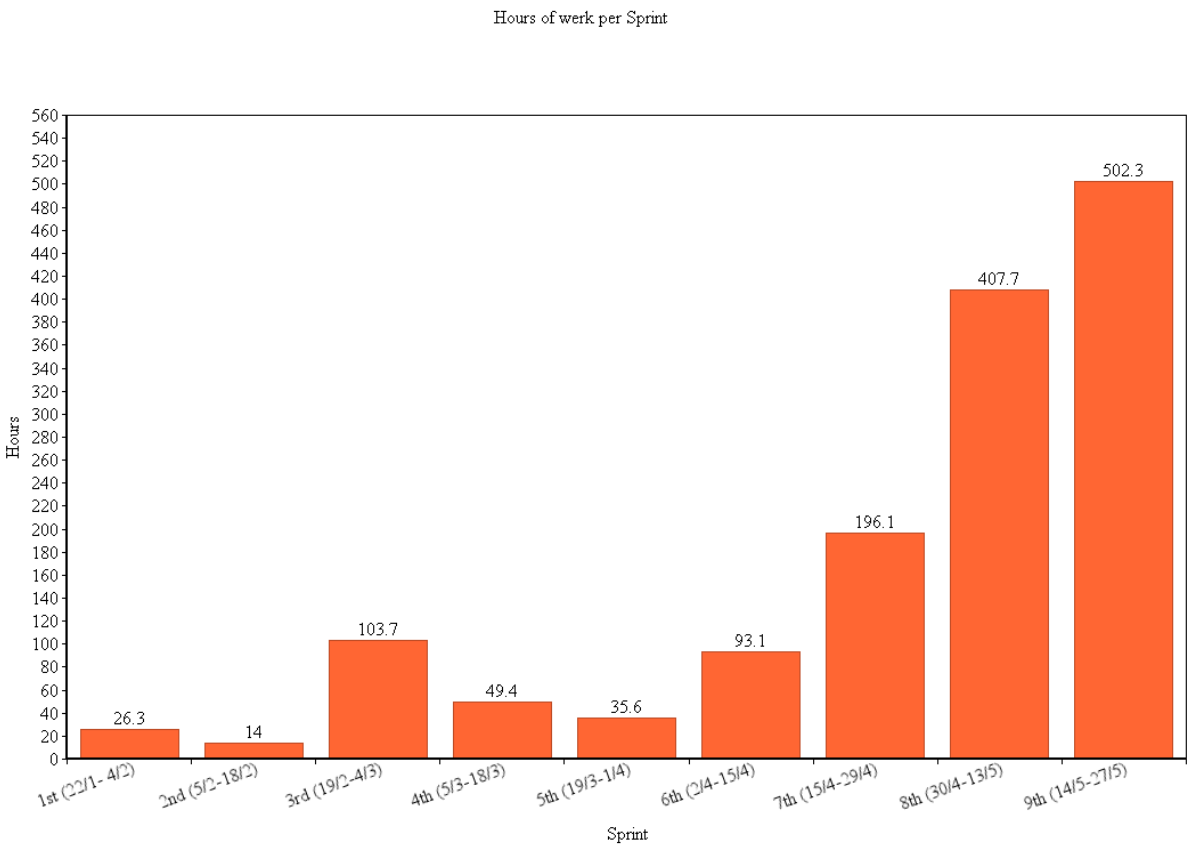
Android Studio: Some of us used Android Studio instead of Eclipse, because iIn Android Studio we could design the interface and make a short video for the Beta presentation.

3.2 Time Management



Created by David M. Szabo, 18.05.2014, Burndown Chart

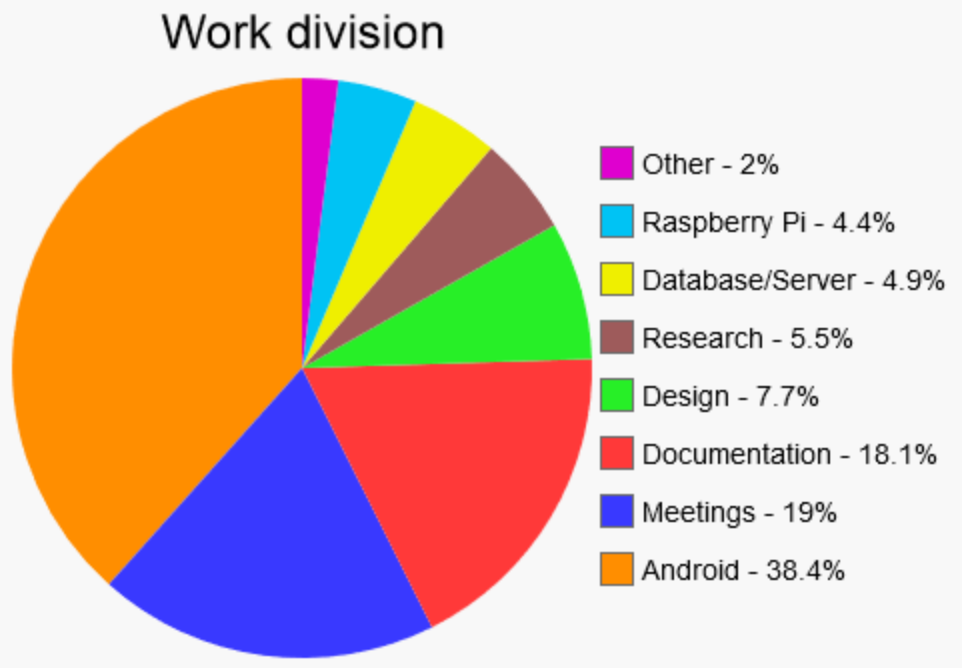
In this chart, from pivotaltracker, we can see that the target deadline was set a bit later than may the 27th that is why we had to work a bit more to reach the project deadline and we put more work in the project in the middle of the term. This can also be seen in the diagram below which illustrates how many hours we have spent working on each sprint.



Created by Henrik Edholm, 26.05.2014., Hours of work / Sprint

Our idea was to do the design documents and coding iteratively. This would mean to design the first 10% we want to implement and then start to implement that 10%. However, we could only manage to do this with the first 10-30% of the features. Then we had a basic design and we started coding more. Then we created the design document after we coded features. The reason for this is that we had the core design structure, but we didn't know how exactly the features will be implemented.

A lot of things created delays in the project work. The first thing is that we had a lot of schoolwork beside the project and we had to concentrate on Logic and Algorithms course and Technical Analysis and Design. The second thing was, for those people who didn't have an android phone it was impossible to implement Google Maps. This delayed the people for about 2-5 days, because they tried to find work-arounds and researching a lot about the different kind of problems. The third big issue was to set up the android environment in Eclipse. For those, who used Linux it was even more difficult and because of that they switched to windows operating system. Thus, using windows and Eclipse or Android Studio did work after all.



Created by Henrik Edholm, 26.05.2014., Work division

The chart is based on the individual work of each team member added together and shows a good representation of our main focus areas in the project. Since we are developing an android application, it comes natural that the most time have been spent there. Throughout the project we have had a large focus on meetings to make sure that everyone knows what's supposed to be done on each day which have increased our productivity. For further insight in every members specific work, refer to Appendix D – Time sheet.

3.3 Group Dynamics

In Scrum there are no groups and no roles, so we kept ourselves to this and tried to educate everybody if some people were working on different problems. However, not everybody was literally coding the Raspberry Pi, but everybody knew in which development phase is the Pi in and what problems are with it.

All of us were contributing to the android development and coded something in it. That can be seen in the code at the code comments.

The whole group was participating in the Sprint Reviews, Retrospectives and Planning meetings. In the Sprint Review we showed the product owner what kind of features we have and what was implemented from the backlog. In the Sprint Retrospectives we tried to come up with 3 positive and 3 negative things regarding the group or project work and tried to resolve the negative aspects. In the Sprint Planning we planned the next 2 weeks and prioritized the backlog with the help of our supervisor. We also assigned the tasks to each member and

estimated how much time it would take with our number cards. The possibilities were: 1, 3, 5, 8 days of work for 1 person.

4. Reflection on risks[7]

4.1 Risks

In this part of the final report we present the possible risks that our team has encountered so far. The risks were divided in the same structure as in the quality plan.

Reflect on project risks – loss of an experienced developer (schedule and resources). Product risk – didn't purchase anything (performance of the software). People risk: required training, key staff is ill, recruiting requirement changes.

4.2 Management risks

Inherent Schedule flaws, delays, underestimation and poor planning

There weren't many managerial problems encountered during this project. Usually, we tried to plan two weeks in advance on every Wednesday and stuck to those plans. Then we revised the plans on the Sprint Meetings. Time estimations turned out to be fairly accurate for many deliverables.

Undefined risks occurring and underestimation

We underestimated how much time it would take to set up the android development tools and this caused delays in programming. Otherwise, we had delays because of the not-project-related school assignments.

Lack of information

When we have felt like we lack some information about various tasks of the project we asked our supervisor or Imed, the Scrum Manager. If we had specific programming problems we started looking for online solutions or in some cases, other groups.

Management change and poor planning

The management didn't change during the project.

4.3 Design risks

Requirement inflation

The risk of having mayor requirements changes and inflation were simply avoided by down-prioritizing the backlog efficiently. For this reason, we have used the time cards and tried to improve the way of our estimations by every sprint retrospectives. We also talked with the product owner every second week in case some of our features are not relevant.

[7] This section references the Naegling Group's Final Report's Risk Section in 2013.

Assumptions about the users

This risk is about making assumptions about the users. We tried to solve it by testing our products and designing it through the android design advices[8].

4.4 Implementation risks

Poor productivity

We solved this risk by having been able to maintain high productivity throughout the entire project by the daily scrum meeting asking people what they have done yesterday. If everyone works on something and somebody is not motivated, then listening to what others have done can motivate people. Also everybody was given a time to finish a task and the person was accountable at the end of time for the given task.

Lack of knowledge

We had a lack of knowledge in android development and python syntax. In order to decrease this risk we had research sessions in these two areas with a definition of done. At the end of the research session everyone had to gain a basic knowledge about the area he or she was researching that was enough to start coding basic features.

When we didn't know how to progress with a problem we tried to search for the answer online and if that didn't help we asked the group members, maybe pair programming as well.

Messy code

The code we produced did work well and even if it at some times looked a bit messy, we did always try to clean it up as much as possible. Commenting what needed to be commented.

4.5 Technical risks

The loss of data

We have been able to never lose any important data by using Github and Dropbox for file-sharing. In Github we have all had our separate branches so that we do not overwrite each other's work.

Technical unavailability

There have been times where some of our group members have been unable to use their Linux operating systems while trying to set up android environment. This has been solved by changing the operating system to Windows and it solved the problems.

[8] http://developer.android.com/design/index.html, 2014.05.18.

Tool technology change

We stuck to the tools we decided in the beginning and didn't use any other. We didn't use Podio, because we thought that the facebook group is enough for messaging each other and keeping each other updated about the meetings.

4.6 People risks

Staff Turnover

Two people joined the team in the beginning of the term and this increased productivity. We could soon enough integrate them in the team, so the integration didn't slow down any project work.

Misunderstandings and conflicts

When there was a conflict or a misunderstanding in the group or between people, we handled it with communicating the problems as a group and individually. We never had any big issues with misunderstanding and conflicts since we all respected each other enough to talk to them when there was an issue.

Staff absence

Staff absence was never really a problem for us, the few times people were sick we usually had meetings online on Google Hangouts. This worked very well and kept the sick people informed what was happening even though they were home getting better. This made it easier for them to jump back in to work when they got back.

Environmental disturbances

As a team, we didn't have problems by others disturbing us. We could always have a separate room, where we worked and nobody really disturbed us.

5. Changes

First, we planned to have a webpage. But later on when we started to implement the different kind of features, and after talking with the product owner we decided not to implement a chat and event flow function. Thus, what we wanted to implement in the webpage the android application already had it, so we decided not to implement a webpage.

Second, we planned to do 10% design in every iteration and then code that 10%. But we had to research how to do coding in android and python and in the last few iterations we were doing more than 10% design and coding.

Third, when the game design group and the python group were done with their part, they joined the android team to implement more features.

6. Appendixes

6.1 Appendix A – Meeting Logs

6.2 Appendix B – Project Proposal

6.3 Appendix C – SCRUM PM

6.4 Appendix D – Time sheet

6.5 Appendix E – backlog

6.6 Appendix F – Workdistribution Naegling