NAO 2K Advanced Home Security System

Software Engineering and Management, IT-section Gothenburg University

Software: Project

Group Ridill

David Lindenberg, Fredric Ola Eidsvik, Flutra Tahiraj, Emil Sundklev, Greg Colbeck, Markus Erlach, Mengjiao Wei.

Github: Ridill-bot

Table of Contents:

1. Introduction 3

2. Scrum PM 3

3 Logs and reports 4

4. Project Idea 5

5. Requirements 7
    5.1 Hardware
    5.2 Software
    5.3 Functional Requirements
    5.4 Non-Functional Requirements

6. Source Code 9

7. Design 12
    7.1 Purpose
    7.2 System overview
    7.3 System architecture
    7.4 Data design
    7.5 Human Interface Design

8. Team member difficulties 14

9. Deadcode 17

10. Functions 18

11. Testing 19

12. Product backlog 20

13. Work environment 24

14. Member thoughts on project 24

15. Conclusion 26

1. Introduction

A system development team presents the NAO robot as your new home security guard. With the new functionality in the NAO robot system, it will be able to detect targets and navigate to it. When the enemy is detected the robot will trigger the nerf gun and be using terminator quote before firing.
The purpose of the project
The idea of this project is for entertaining purpose. A 58cm tall robot that can communicate with humans and detect specific targets where he later aims at and triggers. Just like a terminator. The gun that will be in use will not harm anyone and it's not powerful enough to break anything in the house. But of course with more advance and powerful equipment the NAO robot can be used for more meaningful task. Such as a patrolling house guard that will activate if there is an breakin. where the main purpose is to stop the bugler and call the police, preferably with non lethal means.

Group Ridill:

Markus Erlach
Greg D Colbeck
Flutra Tahiraj
Nicole Musco
Fredric Ola Eidsvik
Mengjiao Wei
David Lindenberg
Emil Sundklev
Scrum-master email: guserlacma@student.gu.se
Platform: NAO Robot

2. Scrum PM:

Scrum is the method what we have been using in our development process throughout the whole project. With a schedule of milestones, events and meeting repeatedly gave us a better learning of leadership. It also gave a good insight of Agile Scrum development and an awareness of the responsibilities that follow with these kinds of environments.

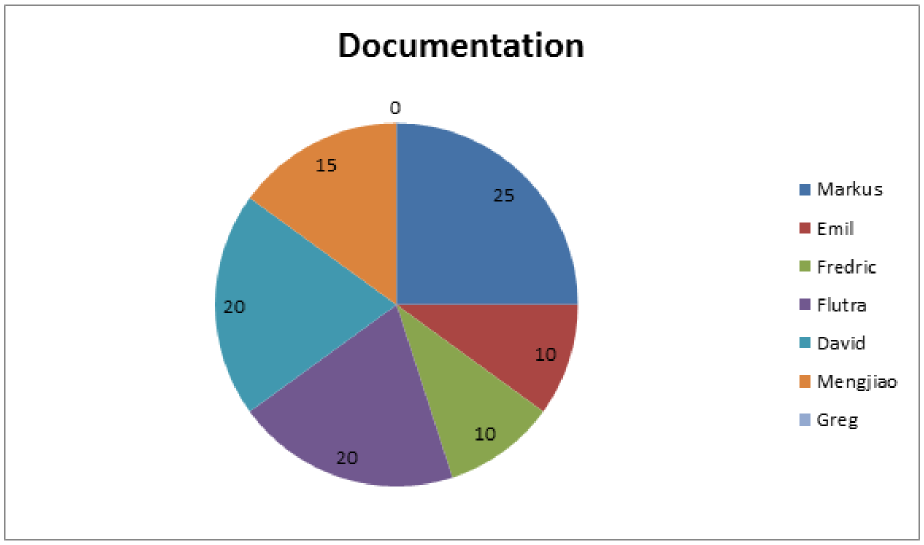
With scrum model in mind and its *three primary rules*, gives us a framework for high productive team cooperation and product development. *The Product owner* establish what needs to be done and manage the product backlog. *The Development Team* will work to achieve these

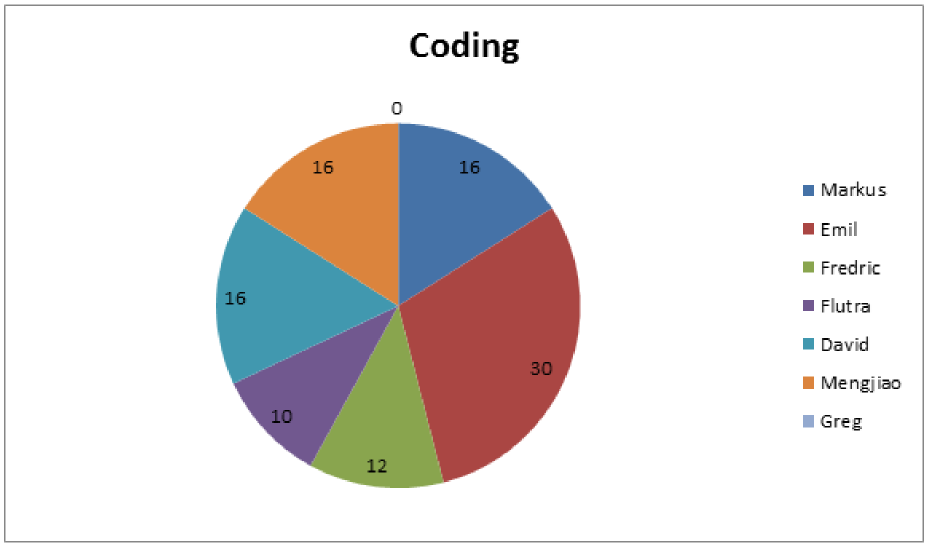
requirements. *The Scrum Master* makes sure everyone understand the goals and follows the scrum planning and sprints. The scrum master will also be responsible for arranging our meetings, be in contact with our supervisor and handle the task management of the group. When we have a good overview of the product backlog our scrum master will then break down and divide up tasks that should be done for each sprint. The development team would then be able to plan out the sprint so that a product increment can be met within each sprint.
Through democratic vote Markus Erlach became our scrum-master and the remaining are the development team. Our supervisor which is Thor Salehi, is also our product owner.
Each morning or meeting in this will consist of daily scrums to make sure we're all up to speed on what has been done and what is to be accomplished while moving forward. After each sprint, our Scrum master should review the sprint with our supervisor to inform on this iteration and give him an update. Then the scrum master should see what needs to be focused on, added, removed or changed for the next sprint.
After the review is done, the development team should incorporate sprint retrospective to inspect itself and see in what or which ways it can improve. The team should see how it can become more effective and increase product quality. We will try to incorporate a burn-down chart so that the group can clearly see what there is left to do and try to keep it updated daily or at least weekly depending on time distribution.
When it comes to technical debt, our team should strive towards the best quality we possibly can while working and at the same time trying to keep to simple designs to ensure that shortcuts aren't used. To ensure this, we will make sure to use our supervisor and keep an open communication with him and also making sure to help each other if needed.

3. Logs and reports:

Everyone had signed up for tasks and because all weren't that experienced with python or programming overall a lot of time went for researching and understanding the language and structure. The project wasn't the only subject that we had to work with and because of that we couldn't only be focusing on the NAO robot. Also with the lack of knowledge and the experience we had, made it harder to know what exactly to research about and we lost productive working hours.
It wasn't till the end of the project that we thought it was a good idea to divide the task among people. Giving them tasks that suits their knowledge.
We had 4 daily scrum meetings a week where we meet up in school in person, and at least one of them was more than 5 min. This was documented. There were times when we used Skype instead of meeting up, but most of the time you couldn't hear each other. Very shortly when we decided to not use Skype that often, only during "emergency", where a meeting was really important to not miss. There were some struggling when it came to hear that person on the other side of the screen.

The work has been distributed as followed:





These graphs are from when we are including our deadcode.
When we only include the recent weeks coding only, Emil percentage will be substantially higher.

4. Project Idea:

What this project is about is mostly for the entertainment, and as mentioned before it's not that powerful at this time.
Before the project could be started, a lot of researches where put in the schedule. Where necessary resources and cost were estimated. When that is taken into care the project was up and running.
The main function was to make the NAO robot entering a duel, where he will walk a few steps,

turn around and fire at the target. The functionalities have been slightly changed because of a huge obstacle that put the productivity at the most lowest point. This led to a turning point, where we needed to change basically the whole project plan, and have a version of the functionalities.

The Network bridge was originally the main thing about our project that was supposed to show our robots short response time. It was intended to reduce the processing of our robot to shorten each calculation thats has to be done. And in return help reduce the time from one command to that command being carried out. The bridge we made worked and we could connect to a server side from a client side wirelessly. We made a few different versions where one could send a string through, one could send a file through and one could send an image through. This along with different encoding and decoding parts we had done made it so that we could almost everything we wanted with our network bridge. The problems we had with the Network bridge all arose late in the project due to our lack of knowledge about the NAO. It only came to light when we were starting to try and put the socket client side on the NAO and we noticed that it does not work.
To fix this we took several steps to try and figure it out. We asked a 3rd year student that had previously worked with this and we were told that they had not figured it out and that they bypassed it by using a java proxy instead. We asked a few different people in our class that we know have previous programming experience and they only told us it would not work due to something wrong on the NAO side of things. It looks for a module called “_socket” that it can not find even though we located it in the NAOs files. So upon realising that we need to adjust our project we began looking for alternatives and found one among the other NAO groups.
They told us about a proxy they were using. Something we upon seeing, decided to implement ourselves to make our project work. This along with the lack of robot time our group got due to lectures, exams and holidays during our scheduled robot time put us quite far behind. Only being able to test our code when we had access to it made it difficult to test or check our code. This set us back hugely and we had to restart our entire project to be able to achieve any kind of progress.
Upon doing so, we had to find a different SDK than choreograph to use. We downloaded NAOQI to get an API to work with and be able to write parts that our robot would be able to obey.

Future Plans – We don't feel that this was an entirely waste of time. We can not use any of this at the current time but for future plans we have considered of making a use of this.

One part of the Network bridge we wanted to implement later on was a live feed through it.
This would help us market the robot as a home security system. So having one part of the network bridge in a mobile application would help us sell the live feed system as well. So when this part of our project got scrapped we were forced to rethink our future plans. So right now, our bot will be a fully automated home system that guards your home. It is meant to make our users feel more safe and at ease whether they are at home our go out. The Nao security system is for anyone who needs an advanced system at home to feel safe. The Nao security system is provides families with a fully automated system with both lethal and non-lethal action that makes you feel safer.

5. Requirements:

5.1 Hardware

NAO robot – A robot bought by Gothenburg University IT-section from the company Aldebaran. The robot has sensors and built in software operated by NAOqi, Aldebaran operation system. Along with the robot there are other programming software tools that are included.

Nerf Gun – A pocket sized blaster that suits the NAO robots hand. Not too powerful to break anything fragile.

Grip Glove – A homemade glove that would help the robot to get a better grip on the gun.

5.2 Software

Python – We chose this programming language because it was the most effective and profitable choice for our position. This language is easier to understand comparing to other language such as C. The structure isn't that advanced and easy on the eyes for beginners. The other reason why we decided to use python as our programming language is because NAO robots is programmed in that language.

Choregraphe – This software development kit will help the developer to create the robots performance. We have been using something called Timeline, which is contained in Choregraphe. With this tool, it gives us the ability to manipulate the robot and store its positions.

5.3 FUNCTIONAL REQUIREMENTS:

MOTION:

1.Make robot walk via voice, programming, shape recognition commands

We would like the robot to be able to walk to certain points by speaking commands. Or that you could have signs along the way and the robot would be able to go a certain route just by following the signs along the way or that we could guide it by speaking commands.

1. Upper body movement (hands, fingers)

Upper body movement is something that we need to be able to control early since we need it. For most of our ideas of what we want the robot to do. If we are able to connect a nerf-gun to the robot, the finger coordination is most crucial and without it we would have to look for another solution.

2. Hand/eye coordination (nerf gun or laser/ need to relay info from sight to hands)

The robot will handle a Nerf gun or laser, robot’s hand will press the trigger when the robot see motion tracking or the specific target. So it should have a good connection between the eye and hand coordination.

3. Make robot talk

Robot should say the exactly words what we write down in the code.

SYSTEM:

3. Show robots live stream vision

This function is so that we will be able to see what the robot is seeing and know what it is reacting to or where it is looking. Also so that, when we implement the Duel, when the “Duel” is won, the Robot can show the James Bond scene of blood running down (to show the robot lost)

2. Shape recognition

This function is to aid in our further developments with the robot. The shape recognition will work so that when the robot sees a shape, such as a square or triangle, it will react accordingly. We chose to have the robot act differently to each different shape it sees.

2. Facial recognition

The robot shall use it's built in camera, or we shall attach a Xbox-Kinect camera, to Make it recognize different faces and shall then perform a certain action/method Accordingly to the implemented method we have been given it.

3. Motion tracking

The reason we wish to use motion tracking would be for our "duel".
We would want the NAOrobot to be able to locate and "fire" a nerf gun/laser pointer at a targeted person or location. The motion tracking will follow the vision of the NAOrobot targeting something particular. Whether it be a person or object.

2. Change LED lights

To make the lights change color depending on something. Where that something could be either a person or an action performed by the NAOrobot.

3. Playing voice recordings (“I’ll be back”)

This function will run when specific action have occurred or when NAO Robot recognize a target, which can be a person or object.

3. Voice activation (low priority) A recognized voice control may be implemented.
Which activates the robot when only this/these persons voice have been recognized.

5.4 NON FUNCTIONAL:

SYSTEM:

Build bridge between computer and robot (efficiency/ using computer processor instead of robot)

A data bridge is to be established between the Nao and the core processor (on the computer) we shall use to compute the algorithms and functions of the Nao. This is to utilize the expanded, far more efficient and superior hardware available via the computer, compared to what the Nao currently possesses. This will allow the Nao to allocate all available resources into physical function, rather than raw computing – allowing faster, and more receptive responses.

1. Testability (virtual simulator for time robot is not accessible)

We will make use of virtual reality (VR), to ensure minimal downtime on development whilst we cannot access the hardware physically. The VR ensures lifelike and reliable feedback and response, allowing us to more efficiently utilise our time.

1. Connect robot to computer network (accessibility)

This non-function is made so we can program the robot through the network. It will make it easier for the robot to move not having a cable connected to it.
1. Documentation
This non-function is the make a journal of what occurs at entire project. Such things as codes, meetings, work time, workload spread, etc.
1. Response time (reducing by using the bridge b/w the two)
We will try to reduce the response time of the robot by making a bridge between The robot and another hardware (processor?), and use it's processor, since the robots processor is not good enough.

6. Source code:

# Filename: proxy.py
- Creates a string ip and an int port to be used within the class
class makeProxy():
- Whenever an object of this class is created it initializes all the different proxies that we need for all our different functions.

```
    def getMemoryData(self, landmark):
    def getMotionTransform(self, currentCamera):
    def setAngleInterpolation(self, names, keys, times):
    def unstiffRobot(self):
    def initiateWalk(self):
    def robotWalkTo(self, x, y, theta):
    def stopWalk(self):
```
-These definitions only returns a function that already existed for the different proxies, but it was easier to define it within this class to be able to call them in other files, so it got direct access to all the proxies definitions. The name of the definitions are basically the same as the original function they represent.

```
    def sayTerminator(self):
    def sayHasta(self):
```
-These two pretty much says themselves what they do, it makes the robot play / say the file that we call it to play.

```
    def subscribeToLandmark(self):
```
-Subscribes to an event that are recognized in the robots memory and tells the memory whenever a landmark is detected.

def unsubscribeToLandmark(self):
-Unsubsribes to the event above so that when a landmark is detected it doesn't fill the memory with any new possible landmark, just a precaution.

# Filename: detectLandmarks.py

- Start the file by setting the size of the naomark and which camera we want to use, create an object of the class makeProxy from the file proxy.py

class calculateData():
- Creates different object to be used within the class

def detectMark(self):
- Loops until a landmark is detected and gets all the data from the detected naomark

def calcData(self):
- Takes some specific data from the detected naomark and creates different objects that represent the correct data to be calculated later. It all depends on that the size of the naomark is correctly stated at the beginning of this file, otherwise the calculations will be incorrect.

def computeData(self):
- Uses the different data from the calcData() function and calculates the robots distance to the naomark

def getResults(self):
- Print out the result from the computeData() class in x, y and z angles, where x axis is the distance to the mark, the y axis determines if its to the right or left to the robot and the z axis is the height of the naomark

def robotWalk(self):
- Takes the calculated data from the computeData() class and uses that data to make the robot walk 1 meter in front of the naomark. The robots walk pretty accurate accordingly to the x and y axis, but he positions himself pretty bad since the floor was very slippery and make him turn when we didn't wanted him to. We didn't get the z axis to be used for the walk since we didn't fully get the math right for him to turn correctly towards the naomark. Again this was hard to get right since he was slipping on the floor a lot when walking and turning. We tried it on a rug with a sticky surface but the robot is not meant to walk on such a surface so he almost fell when walking, however he placed himself very accurate.

def back(self):
def andForth(self):

- These functions we made to make him walk away from the mark and then turn to face the mark, kind of like a wild west duel. however we didn’t havethe time to fully complete that scenario.

```
    def shoot(self):
```

- Simply a function that we used to make the robot raise his arm while holding the gun and pull the trigger of the nerfgun a few seconds after the arm was raised.

# Filename: takeGun.py

- A simple file that has a class with a function that will raise the robots arm, open his hand and close it to a certain point, just enough to make him grab the gun but not to hard to make him squeeze the trigger too early. This was probably the most difficult thing with the robot, since he he so weak and cannot pull the trigger if we didn’t help him by putting a rubber band around the trigger to remove some pressure to make it possible for him to pull the trigger enough to make him shoot the nerfgun. Again he was so weak that we had to pull the rubber band hard enough to make sure that he could pull the trigger, but the pressure got down so much that he couldn’t grab the gun and walk with it without either dropping the gun or trigger it too soon.

```
class recieving():

    def gun(self):
```

# Filename: shoot.py
# Filename: walkFunctions.py

- These two classes needed the calculated data from the file detectLandmark.py, however we couldn’t get around with extracting the results of the calculated data that gave us the distance from the robot to the naomark, so we simply got around that by placing the functions within the detectLandmark.py and in the calculateData() class.

# Filename: tester.py

- This is the file where we tested the code and executed all the fucntions in the other files and classes to create the scenario that we wanted.

7. Design:

7.1.1 Purpose

The purpose of our software design document is to structure the project and get a clear overview so that we can easily navigate through the project.

7.1.2 Scope

Choregraphe is the name of the software that the robot is using and we are going to write Python for the code, we will do this in sublime text 2 and use an external computer to speed up the program since we learned that the processor in the robot is too slow to have a flowing program. With our software we will have the robot being able to recognize shapes/persons and coordinate the robots body and also program full series of actions for the robot so that the robot will be able to perform a full program of action.

7.1.3 Overview

We will be keeping a record of every meeting during the project, and we will also have a work diary to keep track of what's being done. This will be done to follow the work progress, this is a must for the Scrum module. It allows us to track work efficiency and to optimize the progress of the project in retrospect this helps for changing the backlog and keeping time schedule.
Our team name is Ridill, we consist of 8 members plus a project manager. Ridill uses Scrum as our software process, this provides flexibility and fixed points of completion.
We have Markus as scrum master that will control the sprints and meetings. The other share the work load as programmers, testers and documenting.

7.1.4 Reference material

The material that we are using is http://www.aldebaran-robotics.com/en/ for the robots program. And we are using Python for the code http://www.python.org/.
Rhapsody will be used for the modeling of the project
http://www-03.ibm.com/software/products/en/ratirhapfami
Pivotal tracker will be used for tracking sprints. https://www.pivotaltracker.com

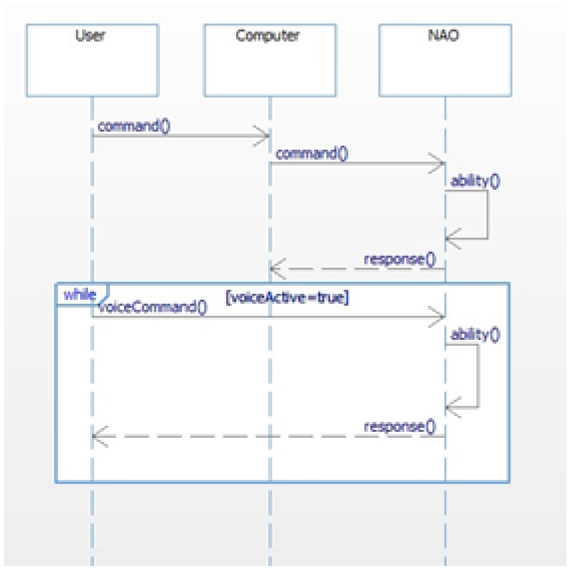
7.2. System overview

Our project goal is to program the NAO robot to be able to entertain an audience by doing certain tasks that are preprogrammed. Our main objective for the robot is to do, a duel. To achieve this duel, our robot is supposed to be able to track objects (motion tracking), walk around, lift its arm, calculate distance and track the “object” in front of it to fire a nerf-gun or laser.
Other tasks for our robot to achieve is things like, object recognition, communicating via external speakers and a built-in microphone. Count things it does, like amount of steps it takes in the duel.
Other small tasks we wish to accomplish, is making the robot tell or recognize jokes, have a built in voice activated menu that activates upon startup.

7.3. System Architecture

7.3.1 Architectural design

To design the architecture for the design we have to put it in modules according to a white box model. The user, the robot and a computer are the classes we have in mind when structuring this. Firstly, we want the robot to be able to interact with our user through the computer. The robot will see or hear something that gets interpreted by the robot, which will then interact with the user.



The user will also be able to input commands into the computer which will make the robot perform the expected commands. Perhaps a command is input so that the robot moves or speaks. A command should also be possible to be spoken and therefore skipping the computer interaction part of the model.

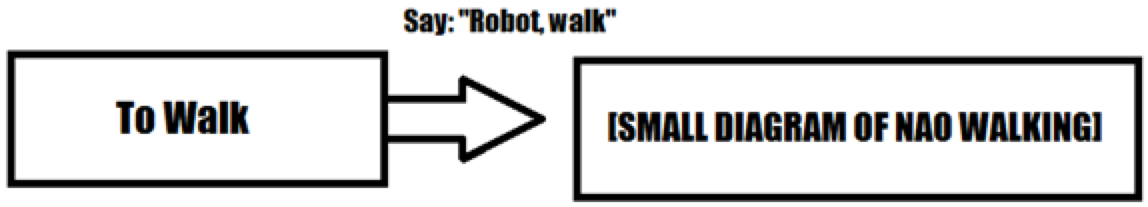
7.4. Data design

7.4.1 data description

We will need to create a data base. Most likely we will be using SQL. It is the database we have most knowledge about and experience with. We will use external storage for storing data since the robot can't store data as far as we know. It is also meant to save the robots capacity and efficiency.
Most likely we´ll be needing a database to store pictures or objects we put in for our recognition algorithms and to make it easier to save data for our robot to interpret. The database should also store sounds for playback.
The database will be stored locally, as the only reason it needs to be accessed is locally via the NAO.

7.5. Human Interface Design

Our primary focus for the NAO is not to have a GUI upon a screen, but to provide the user with a sheet of information on specifics actions and commands that are available for the NAO. On this sheet, there will be basic instructions, followed by a basic diagram of the outcome:



We intend for all of our feedback to be physical and kinetic, not relying on a GUI to relay information for the *user.*

8. Team Members Difficulties:

The environment among the team members have been rough in different levels. Have been involving losing team member, members health have been on the stake, structure plan have been changing constantly. This has cause a lot of stress and irritations on everyone. Members not showing up in time without any advance notice to the group, members not showing up at all, members not taking responsibility that are expected from each one. This sets the rest of the group in a difficult position to re-plan schedules and trust in the group became questioned. The frustration resulted into work putting away, to try and fix this issues among the group. So this was very important to solve, because without a working communication between the members the problem will remain and the project will never be able to be completed.

We had a penalty system that everyone needed to follow, and the punishment weren't that rough. For every 4th mark, you would get to bring cake or coffee and the penalty would be enforced. The 5th penalty would be to take the problem over to our supervisor. Even though someone got that 5th penalty we gave that person new chances. Maybe that's why people stopped to really care, but things got worse and we arranged a big meeting that only would be about this topic. Unfortunately there was too much aggression between two members so one of them made the choice to leave the group. Now with one team member sort we needed to put extra work on someone else shoulders where the time of working hours were limited. If the communication would improve from this point it would have been worth the extra work and stress, but now that wasn't the case. The communications were in a stable stage but everyday we noticed some slack of work from some members and it was only matter of time until we were at the same situation. We have tried several times to approach this problem in different angels sadly the issues got worse. This brought out the bad side of everyone.

Team Members:

8.1 David – Rarely absent at meetings and also rarely late more than 15 minutes, which is in the "academical quarter" and was acceptable. Always told the group in advance when he had work to do outside the project. His working method was most of the time in its best when he were multi-tasking. That didn't always mean two things about the project, sometimes it was playing the game“Hearthstone” and sprint work. This gave the *impression* of boredness or just no participation, but it fact this was working for him, and for us because in the end the work got done.

8.2 Emil – In the beginning of the project he was always on time, but when problem started to show up among the team members he started to slack. When present he delivers great work, but we needed to wake him up, sometimes more than once. It could take him at least 30 minutes when it should take him 10 minutes. There have been occasions where he never showed up because of oversleeping. This was very frustrating for the other team members, because we saw signs of a repeated issue-history which made us question if work would be done if we let him work from home. It did but it was like walking in the dark till the next day because there wasn't any updates on our Facebook-group where we are supposed to inform and update everyone about our working. This was partially a technical problem for him but there was ways around it.

8.3 Flutra – Rarely absent at meetings and by choice in place well before the meetings and gave the group absence notification in advance. Had hard time when it came to researching topics, which in result reduced working hours. The software part needs to be improved some more. Likes to understand what others do and try to help. Maybe better to focus on one's own part but when you are on an endpoint and there’s not much you could do. Help your team member or go home, and sometimes home was chosen. Not always what others thought was okey.

8.4 Fredric – There was a time when he was absent from the meetings a couple of times, but there were work related and posted in advanced. But with all tension that already was created in the group, things like this when you're not present could cause irritations on some. He was a bit all over the place, a great person to be around with and lightens the mood but topics got easily of track. But when the focus was at its top he had many potential solutions to share.

8.5 Greg – Few times when he is present and on time. Lack of participation when it came to the project but also other school work and casually absence caused frustration. We had to call him every morning because of the sleeping problems he had. Although he had been called to wake up he could come in late. This wasn't okey because members had to wait and meeting had to be postponed. Most of the time during meetings or group work session he tried to look for apartments and other outside school stuff, this could wait and instead he could have shown bit more effort in what was going on in the project. We didn't see any change and we got tired of it. This was also a big frustration in the group because information did not reach to everyone because of the lack of focus the group had and caused to another. This became a big issue in the group and working hours were being lost every day. We reached a point where aggressions were spoken out verbally and written, a big meeting was arranged. Our former member Nicole, made the choice to leave the group because of personal reasons and she didn't see any future change. The issues were put on the table and what the group excepted from Greg was a medical certificate, updating the group from home about his workload, and letting the group know in advance when he can't attend meetings. This was followed badly, most of the time one member got the information or no one did. Greg have been to many doctors appointment and it's understandable that coming in for meetings was exhausting but the time came where we had to take this issue to someone with bigger impact, our supervisor. The communication between Greg and the group died and weeks went by without hearing from him. At this point the group felt that we've done our best to approach the problem but with this response we left it to the supervisor and Greg himself. Two working days left till this project ends and the status is the same.

8.6 Markus – Rarely absent, when he is, it's because of sickness absence. Work gets done both during group work sessions and from home. His scrum master roll needed to be improved, especially during the time when the group was under a big stress, both because group issues and sprints getting delayed (not entirely the groups fault, missing SDK). When the problem was taken to our supervisor, a weight of worries and lack of trust lifted away from our shoulders and he no longer needed to push each individual make sure all the work was getting done.

8.7 Mengjiao – Rarely absent and in time. Working on her sprints and always gives a try to help the one needing.

In general the group can take breaks where we play games and try to beat each others score on the board. This has been handled in a good way, and it helps ease the stress of all for one moment.
The communication in the group have improved a lot and the productivity along with it.

9. Deadcode:

For our project we have quite a lot dead code because we started out making a socket server and a client side that was intended to be a network bridge. We couldn't use this so this will be a part of our deadcode. Also we have our speech recognition code for the robot since we didn't manage to finish it in time, same goes with the LED code. We never implemented it for the same reason, lack of time after having to change from using choreograph to only python. It’s not optimal to have this much deadcode that we never finished but it all comes back to the fact that we started the project by making the bridge that later on was discovered to be unusable and not possible to implement..

When this was discovered we had to change the project and update the backlog and the focus became to remake the project into python code so we still had a project. But the amount of work hours we have put down on this code and what we have accomplished with it is still something we want to report.

We also have code to create a simple database using SQLite. It basicly only creates a simple database to store information such as text and integers.

The Network bridge is fully functioning and can be used to send strings or files through. It also encodes and decodes whatever is sent through to make it even possible.
The server side for sending files, creates a socket on the server side of things on the desired port and ip you choose.
It binds the address with you ip and port and then listens for incoming connections.
When a connection is made, it waits for incoming data and decodes it. Once it is decoded it prints out the name of the received file.

The client side of our network bridge for sending files is also working. It connects to the IP and Port of your choosing and attempts to send a file to that address. It reads in the file you want to send, encodes and sends it. It prints out the file being sent and then at the end before closing, it also prints out what has been sent. For example, if a text document is sent, it will print the text inside the document.

One version of the bridge is to send an image and recreate it. this was created for the purpose of the robot using the camera, the idea was to send an image to the robot and then the robot was supposed to take that image and scan the room for the same image and use that as a target. The version of the bridge that we have since we didn't finish it, because the robot couldn't use it. Is that we requests an image though the server from the client. the client sends an encoded image to the server side. Then the server decodes that image and creates a new replica of that image. The idea of this was to make it so that the robot scans the room taking pictures of what it sees and sends that encoded to the server to keep up the speed of the program and then the server with a faster processor decodes these pictures and sees if there are any of the pictures containing our "target" which should be predetermined.

10. Functions:

A use case diagram has been made with our documentation and added alongside this to show our project and explain in steps how to use our different functions that are explained below.

10.1 Walk:

The Walk function is made to just be able to have the NAO robot move around.

10.2 Detect:

Landmark detection is one of our main functions that helps us be able to use the position function, the shoot function and is also core in making the robot being able to duel(not fully functioning for now). The detect function makes the NAO scan in front of him to locate a landmark. If a landmark is detected, the NAO will print out the landmarks location in relation to the NAO.

10.3 Shoot:
The shoot function is there to make the NAO be able to shoot at a detected landmark to be able to eventually duel a person or landmark. The NAO raises his arm and slightly opens his hand to be able to “squeeze” the trigger and shoot.

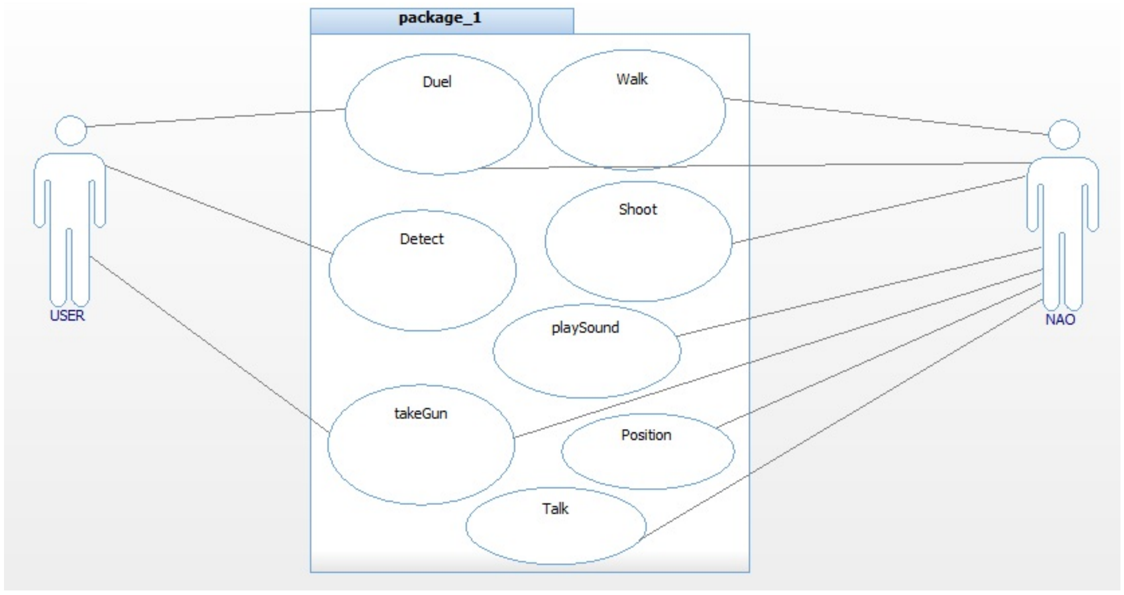
10.4 playSound:
The playSound function is here to take use of and play up added .wav files in the NAO. Specifically playing terminator sounds to make it more entertaining to outsiders and at the same time making it more enjoyable for us in the group to work with.

10.5 Talk:
Making use of the built-in talk function of the NAO, we can make him say command that we input. This just adds another layer to interacting with the NAO and is quite useful for when coding to be able to check whether or not a function is working.

10.6 takeGun:
This is just a simple function that we make use of in order to be able to carry and shoot the nerf-gun in the other functions. The function is activated after the landmark have been detected and the robot raise his arm and opens the hand, this is when we hand the NAO the nerf-gun and the robot closes the hand half way to be able to carry it but not to shoot..

10.7 Position:
This functions take the detected landmark coordinates and positions himself 1 meter from the target to be able to shoot at it.

10.8 Duel:
The dueling function makes use of all the other functions to be able to simulate a duel against a person or a landmark.The NAO detects a landmark, prints the location, takes the nerf-gun from a person. The NAO then positions himself a meter from the target, takes aim and fires.
Originally we intended for him to turn around and walk away at first when a landmark has been detected. Then turn around again to face his “opponent” and fire the gun. But due to time constraints and other technical difficulties we have had to simplify our current “duel”.



11. Testing:

At the beginning of the project we were working with the bridge and we could perform a lot of testing on our own computers. With that we could see that the bridge was working both on the client side and the server side, but when we had access to the Nao robot we discovered that it wasn't working and after having worked on the bridge quite a long time without finding a solution on how to get it to work on the robot we realized that we can’t use the socket server on the robot. This forced us to switch the way we were programming the robot. The solution became to write everything in python, with the python code we can run it on the robot and the testing with the robot is working. This however caused another problem, now we can’t test the code without the robot. This made us work with our code in the blind the only thing we could test without the robot was if there where python syntax errors or indent errors. it became a little bit of a challenge because several times we managed to get codes that we thought was working, since it didn't give us any errors but when we got access to the robot it became clear that it wasn't working with the robot. So we spent a lot of time trying to correct the code with the robot and when we left for the day it felt like we hadn't accomplished anything new or got any further with the project. instead it felt like the whole weeks progress had been for nothing because we had spent the day we actually had the robot to rework what we had done during the week

Another struggle we had with all the coding in python, was that we needed to code using an old version of python, 2.7. This also made it harder to get information and made us spend more time converting or trying to figure out how to use old syntax since we had previously only been using the newer version with the network bridge.

12. Product Backlog:

		Product backlog	
Story ID	Priority	Description	Story Points
1	high	Socket connection module(server) Functional requirement is open socket, close socket and pass on the socket.	2
2	high	Socket connection module(NAO) Functional requirement is open socket, close socket and pass on the socket.	2
3	high	Encoder(NAO) Functional requirement is encode to selected language and no data loss.	1
4	high	Decoder(NAO) Functional requirement is decode correctly from selected and no data loss.	1
5	high	Develop decoder/encoder language Functional requirement is choose language , structure semantics and transmittable.	2
6	high	Encoder(server) Functional requirement is encode to selected language and no data loss.	1
7	high	Decoder(server) Functional requirement is decode correctly from selected and no data loss.	1
8	high	Transmit module(server) Functional requirement is transmit data, no data loss, use received	1

		socket and receive data from encoder.	
9	high	Receiver module(server) Functional requirement is receive data, no data loss, use receive socket and send data to decoder.	1
10	high	Transmit module(NAO) Functional requirement is transmit data, no data loss, use received socket and receive data from encoder.	1
11	high	Receiver module(NAO) Functional requirement is receive data, no data loss, use receive socket and send data to decoder.	1
12		Central LED control Implement all LED functions and create an LED control. Functional requirement is receive order in selected language format, and translate order to appropriate LED action. Glow,Colors The ability to glow sporadically, in a set routine and on command.	2
13		Assemble the bridge: connection The connector Transmit modules and the connection modules to work together and transmit data back and forth. - Physical proof that there is data being sent back and forth.	3
14		Record sound To utilise the NAO's on-board microphone to record sounds and ad them to the sound library for retrieval later. -Record Sound	1

		-Give specific name	
15		Head sensor The ability to react or act in a certain manner when interacted with.	
16		Play sound To draw upon the sound library on command or within a set routine. The sound retrieved could be random, or a specific set sound depending on situation.	
17		Walking The ability for the robot to walk a pre-defined path, a set distance, or on command.	
18		Speech recognition(unfinish) The ability to perform a specific function or react in a certain way depending on which specific voice spoke the command. Different voices produce different results.	
19		Set language Set the desired language we want the robot to activate with.	
20		Shape recognition(unfinish) The ability to recognize rudimentary 2D shapes (square, circle, triangle), whether they be physical objects, or shapes drawn on a surface.	

Sprint 1
goals: 16 points/ 1 chore
design and implementation of the network module
achievement: 0 points/ 0 chores
conclusion: sprint failed

Sprint 2
goals: 16 points/ 0 chore

achievement: 10 points
conclusion: Sprint successful

Sprint 3
goals: 15 points
sprint achievement: 10 points
databridge assembled - 8 points
com protocol- 2 points
conclusion: partial success

Sprint 4
Sprint Goals 21 Point / 0 Chore

Sprint Achievement: 10 Points

Sprint 5
Sprint Goals 21 Point / 0 Chore

Sprint Achievement: 10 Points

Sprint 6
Sprint Goals:
Finalize the project (20 points)

Sprint Achievement: 25 Points

Sprint 7

Record the robot

We forced on building data bridge during 5 sprints, we spent a lot of time to do the research about how to build data bridge. Unfortunately, we can not put the server on the robot. That means we have to change entire plan. So at the beginning of sprint 6, we decide to use naoqi SDK to connect robot with computer, then we finalized the project. Our supervisor decide to spend two intensive weeks to finish project. At the end of this sprint, we have done 5 points more than our goals.

13. Work Environment:

For this project we as a group needed to work with a few new tools that we previously had no experience with. This caused a lot of ramp up time where we needed to focus on researching

the different ways and tools we needed for the project and then also trying to learn network programming.
Having to work with Choregraphe was quite easy at first, but then when we needed to change to almost only python code, it all got a lot harder due to the difficulties we encountered when trying to export Choregraphe timelines to python. These problems and the problem with Aldebarans website and actually locating a special SDK we needed slowed us down quite a bit.
The special SDK we needed, we eventually received from another group that had previously downloaded it. When we logged into the Aldebaran website we could not locate the needed file and had to ask the other group for it.
Another issue we had when it came to getting work done was that the lack of time with the robot hindered us from progressing from time to time. If we thought we were done with a part of code and needed to test it, we had to wait until we could use the robot to be able to do so. The testing was also needed so that we could make that we on the right track. Without it, we could never be quite sure the code was actually done. This caused a lot of unnecessary down time that probably could have been avoided with perhaps better planning on our side and the person responsible for the robot.

14. Personal thoughts about project:

Markus: I personally am not a fan of the NAO due to a bad working environment and not being able to find and use what we need. A part from that, I liked the actual time we had with the robot, it was quite fun to work as a group to accomplish a program that made the robot do what we wanted.

Mengjiao: the big problem is we dont have a lot of time with robot, we share the robot with other groups/we only have time work with robot like 3 times a week. It is not enough for our group, we have to write in python code. We have to work with robot to test and fix code. It is really fun to work with this robot, but i think if we have more time we can do more features and have more functions.

Emil: My thoughts are mixed, it was fun to be able to work with a robot for the first time, but then the robot weren’t a very good robot, and i really think if the robot didn’t get hot all the time after a short time of usage, that the battery didn’t die so quickly, and if the robot could have been able to pull the trigger of our gun with a little more power, things would have been more fun in my opinion.

Flutra: To be able to work with the NAO robot was interesting. There were a lot of obstacle that the group ran in to. And I think everything would be much more fun if the robot was easier to work with and our communications was as good as it got in the end of the project. I know now i will not chose NAO again at least.

David: The project faced us with a big challenge, it was fun to have such an open project where we could decide a lot for our self. But the knowledge about the robot made us spend too much time of researching and trying to find out what we could do and not. there were also no source that we know had the information. We did use the Aldebaran API which helped a lot but was not very clear on how you were suppose to write the code. It was more of an guideline. If I were to do this project again I would avoid the robot if I could but if i had to use it again I would demand more time with it. not as much longer days as being able to test the robot when I needed instead of having set days. It's okay for the robot to have to be in a certain room but we would need to have access to it everyday. Also there should be someone that have knowledge about the robot that you could ask for help. Not just a supervisor but an actual person with deeper knowledge about the robot.

Fredric: I did not like working with choregraphe and i did not like the working hours because it was too little time. However when we started working with python it was getting more exciting and we start getting result. So then it was fun and very educational. The working environment did not have enough power outlet and was very messy all the time due to the other projects going on.

15. Conclusion:

Working with this project have given us a lot of challenges and a good view of what working on a software project could be like. First of all one thing that became clear to us was how important it is to have good knowledge about the hardware and software that you are using is. The main example that we have of this is the bridge, when we didn't get it to work on the robot we spent a lot of time researching about the robot and about python to fix the problem. When it later became clear that it wouldn't work at all we had spent a lot of time on this for nothing. if we were to have more knowledge about this or if we had been more comfortable about python this might not have taken so long, and we would probably had discovered earlier that wouldn'tent work. This would make us to change the form of coding we used earlier. We did switch to wright the whole project in python, and used a proxy instead of the bride to connect to the robot. We did use the function timeline in Choregraphe to store positions for the robot that we needed for the aim and to make it smooth when handing NAO the nerf-gun. Because we spent so long on trying to fix the bridge we were lacking time once we switched to the proxy, and we experienced that it was really hard to create thwholele project again since we needed the robot to try any code and the lack of time we got with the robot really became an obstacle. However the program we manage to create in the timespan we had is almost more than we could hope for. We got all of the main features that we were hoping for, although we were hoping to extend the features to make the program more advanced and accurate according to our first backlog, where the robot would be able to use speech recognition and grab the gun for himself. Due to the late restructure of the project, we were unable to follow a quality assurance plan, and we strayed a bit off our Scrum structure and more focused on getting all the functions done. If we were to have more time we would have updated our backlog and split up the tasks so that everyone could have worked on their own part but since we couldn’t test the code without the robot we were having crisis meetings where we all helped each other writing the code. Later when we had the robot we had to test this code but it didn’t give us much time to work on new code when we actually had the robot.