Software Engineering and Project Management (1DL251)

Final Thesis

25 October, 2020

Group F

Anton Bergåker

Anwar Chowdhury

Carolina Gripsborn

Nusrat Hossain

Anto Juko

Catya Kemppainen

Marie Paulsson

Srinivasa Sai Akhil Raghupathi

Sandeep Sunny

1. Scope statement

This document presents the final thesis for developing a terminal-based two player board game. The game, called UU-Game, is developed for a small russian gaming company. There are 3 main components of the game. Game engine,communication platform and Game platform.

- A **communication platform** to configure tournaments for 2-player games to play locally and on-line. They must be able to choose between a casual local/on-line game against another player, a casual local/on-line game against the computer or to create a tournament. The tournament option must build a schedule for up to 8 players, start and stop the games when needed and keep track of players' classification.
- A UU-GAME **game platform** is the game interface. It must allow human players to play a local UU-GAME. That is, it must show the game interface (e.g., the board, the pieces already placed on it) and control the game state (e.g., who must play next, if the game has finished) and verify the outcome.This component acts as the main menu of the application, where players can pick what type of game they want to play.
- A **game engine**, able to play the UU-GAME. This component is very similar to the *game platform*. However, instead of acting just as a game manager, it must be able to give a specific game state and play the next best move in the game.

This document includes the details of design and integration and testing plan of the game. In the design part it describes the design of our own component that is the game platform and in the integration and testing plan it describes the integration of our game platform with other two purchased components named communication platform and game engine.
A lesson learned section is also included where we describe what went well and what could be done better. Also we included all the meeting notes,agreements of team,contracts etc.

2. Design

2.1. Design Scope Statement

This design part outlines the requirements for a terminal-based game platform that allows players to play the board game UU-Game. Players should be able to choose between different game play options in a menu and through writing commands in the terminal move pieces on the board and play the game.
The main menu should provide the players with options of how to play the game (e.g. online/local) through printed text in the terminal. While playing the board should be drawn as a picture in the terminal that shows the players

what the current playing board looks like. The players interact with the game platform by writing commands in a specified format in the terminal which are interpreted by the platform and the appropriate action is taken.

2.2. Definitions

2.2.1. Node

A node is an intersection between lines on the gaming board where the players can place their pieces during the game. Only one piece can be placed on any one node.

2.3. Components

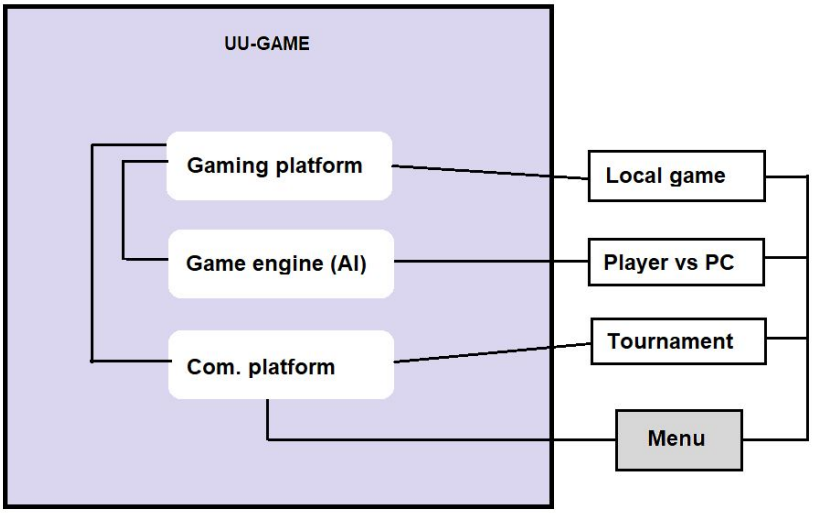

Figure 1: Component diagram

2.3.1. Game board

The board is a rectangular shape with intersecting lines. The intersections are nodes and the board has a total of 24 nodes. The game pieces are placed on these nodes during the game, see Figure 2.

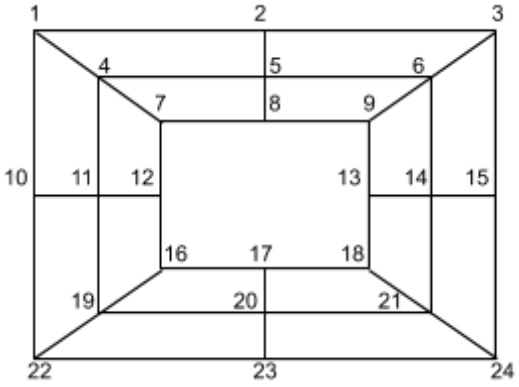

Figure 2: The UU-Game board

2.3.2. Pieces

The pieces are one of two colors: black or white. Pieces are placed and moved around the board by the players. They can be placed on the intersections of the lines (nodes) on the board. Total of 24 pieces, 12 pieces for each color.

2.3.3. Start screen (menu)

The start screen (menu) presents the players with the options “play local”, “play online”, “play against AI”, “How to play” and “quit”. The player types their choice in the terminal.

2.3.4. End Screen

On the end screen the players can see who won the game. Additionally, there will be “play again”, “menu” and “quit” options for the players to choose between.

2.4. Functional requirements

2.4.1. Player

1. A player needs to be able to choose a game mode (online, local, vs AI) to start the game.
2. A player needs to be able to give terminal commands that are interpreted as different actions in order to move pieces on the board.
3. If a player has three pieces in a row they must be able to remove one of the opponent's pieces to try and win.

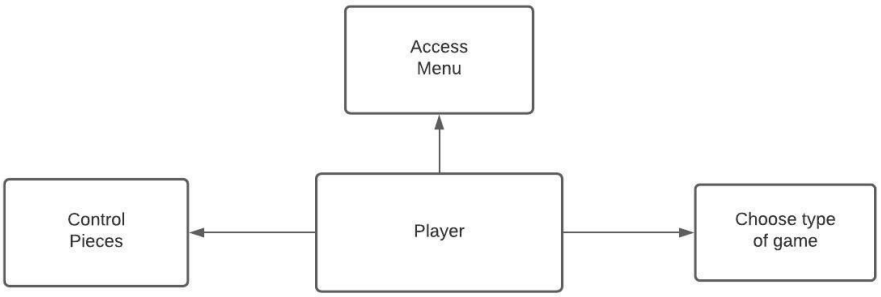

Figure 3: Interaction diagram for Player

2.4.2. Playing a game

When the player starts the program, they will be presented with the main menu. The menu gives the player choices of how to play the game (local, online, vs AI) or to quit. After the player makes a decision through the terminal, the game board is drawn. The player makes moves by specifying placements/movements of pieces in the terminal and the board is updated accordingly. When a game is finished, an end screen will then be displayed stating the winner and providing options of what to do next (quit, play again, main menu).

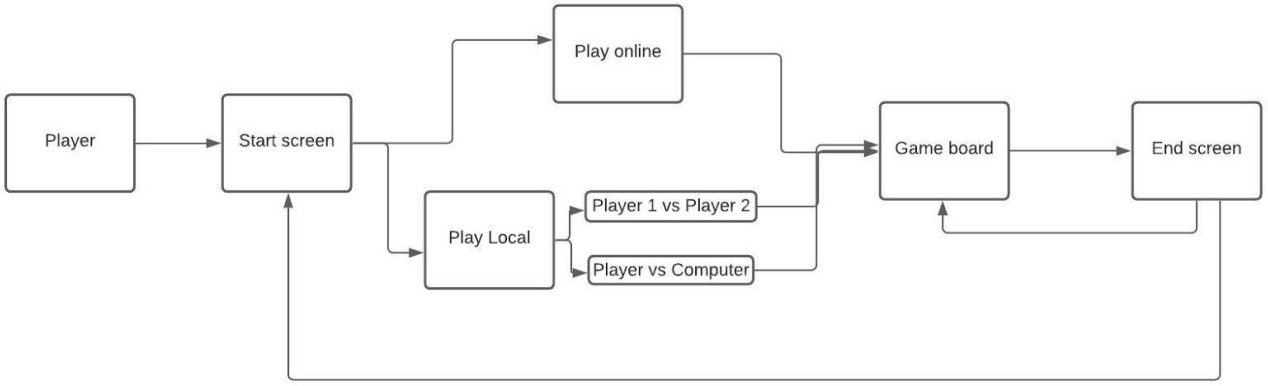

Figure 4: Sequence diagram of game process

2.5. System constraints

To be able to use the UU-Game, the user must have Python version 3.9.x and have a working keyboard.

2.6. Non-functional requirements

2.6.1. **The components should be written in Python for easier integration.**

As the game platform is programmed in python, it's necessary to install a version of Python on the computer to run the game.

2.6.2. **The user must have a working keyboard to be able to play the UU-Game.**

Different key inputs are necessary to play the game so it's essential to have a keyboard to be able to play the UU-Game.

3. Integration plan

3.1. Introduction

This section describes details of our integration plan for a communication platform and a game engine for a UU-Game with our game platform.

After managing to sell our component our budget allowed us to be able to purchase both of the remaining components to complete our project. For both of the contracts there is integration support from the contractor included, which simplifies the integration process for us. For the game engine we are going to be working with developers from the contractor during the whole process as they also purchased our game platform.

3.2. Integration plan

3.2.1. **Game engine and Game platform**

A Game engine for the UU-Game will be provided by Group C. According to the contract there will be two developers from each team cooperating with the integration of the two components, as Group C also purchased our game platform. There will be 2-4 scheduled sessions per week for the integration.

Main tasks:

1. Translate the game engine's board nodes so that it is a 1-to-1 representation of the game platform's board nodes.
2. Read and write to a JSON file of the game state to and from the game engine.
3. Make it possible to initiate an AI with a chosen difficulty to play against.

Activity table 1

Task	Effort (%)	Duration (days)	Milestone	Dependencies
T1	50	0.25		
T2	75	0.6	M1	T1
T3	75	0.5	M2	T1, T2 (M1)

3.2.2. Communication Platform

A communication platform for the UU-Game will be provided by Group I. According to the contract there will be some support for the integration from the contractor. One or two developers from the contractor will be present at a scheduled meeting and go through their most important functionalities and what might need to be altered to work for our component.

Main tasks:

1. Add all the communication platform menu options to our main menu.
2. Send a JSON file with the game state between players.
3. Set up a vs AI game if competing with an AI in a tournament.
4. Start a game between two connected players in a tournament.

Activity table 2

Task	Effort (%)	Duration (days)	Milestone	Dependencies
T1	50	0.25		
T2	75	1	M1	
T3	75	0.5		T2 (M1)
T4	75	1	M2	T2 (M1)

3.3. Justification

Our original plan was to sell our component for 80-120 Uppsalines, however, when researching the market we noticed that the prices were only around 80 Uppsalines, perhaps with an extra feature for a slightly higher one. Therefore we decided to sell our product for only 80 Uppsalines.

We also planned to buy at least one of the other two components, and possibly the other as well if there were enough Uppsalines left. This worked

out as we managed to sell our component, making it possible to buy both of the other components.

4. Testing plan

4.1. Scope Statement

Due to time constraints we decided not to have a test suite for our project during the majority of development. We chose to prioritise the integration of the other components since we purchased both of them rather than implementing them ourselves. Though this creates a risk of certain functionalities not working as expected under all conditions, it was necessary to deliver the products on time.

5. Lessons learned

5.1. Decisions

One of the major decisions of our group was to buy both of the components as we had enough Uppsalines to do it. We took the decision by majority of consent and informed everyone of the team. Also after having meetings with different groups, we as a team decided to buy the component which matched our component the best and also provided the best offer in integration assistance.

Another decision was selling our game platform to the same group that we purchased the game engine from, so that we could help each other in the integration process.

Through this whole process we also learned to make decisions in collaboration with all the members.

5.2. Teamwork

We are 9 people in our group. As this is a large number of people we divided our team into two smaller groups: one group that manages the development of the game platform and another group that manages the documentation, contracts, writing etc. Each smaller group also helped each other when it was necessary. This division was done at the beginning of the project according to each group member's interest. We also assigned a specific role with some responsibility to each member and everyone performed accordingly. Whenever any important decision needed to be made it was done again in a big group rather than any one person’s own decision. Sometimes it took a little bit longer to make a decision but overall it worked well for us as all the

members kept good communication with each other.

5.3. Strategy

5.3.1. Monday Meeting

At the start of the project we decided to have monday morning meetings where we would discuss the whole team work plan, discuss all the problems we might have, and the tasks to be done . Through these meetings we learned that we should have maintained an agenda for every meeting, to make them go smoother and make sure everyone knows what is being discussed.

5.3.2. Github

We used github to keep track of our development of code. All the developers used that whenever it was necessary to update the code so that others can also check what's going on.

5.3.3. Trello Board

This was one of the biggest learning points through this project. We were new to using trello, but we still tried our best to use it by adding the user stories and story points. Sometimes it happened that we forgot to add user stories or update points. Group members also had to sometimes take responsibility and gave reminders to everyone to write the task in trello.

5.3.4. Google drive

We used google drive to share all the documents with each other. All the deliverables, contracts, meeting notes etc were kept in Google Drive. So if any group member missed a meeting they could keep track of what was going on.

5.3.5. Communication in slack

We used Slack for communicating with each other. We had three slack channels, one was a general one where everyone was and discussed things that concerned the whole group, another one was for developers and the last for documentation. Through this project we learned it's effective to have small groups to do small tasks.

5.3.6. Doodle

At the beginning of the project we had some problems fixing time for meetings .There were lots of schedule conflict among all the members. We were discussing meeting plans in general slack which

was hard to keep track of. So after some time we decided to use Doodle to fix the meeting instead. Using a Doodle made our decision making a lot easier, since everyone could just choose the times they were available and then we set a meeting at the most convenient time.

5.3.7. Zoom

The meetings were held by using Zoom. It made it possible to collaborate with each other and other groups. It also gave the option to screen-share which made it easier to collaborate.

6. Annexes

6.1. D1 [Group Agreement]

6.1.1. Group Members

Carolina Gripsborn

- Master's Programme in Computer and Information Engineering

I have used Python before but not to a great extent. I have also designed simple games before however not on a big scale. I have used the Scrum method numerous times before in other projects and feel quite comfortable with it.

Srinivasa Sai Akhil Raghupathi

- Master's Programme in Embedded System

I have taken up one of the courses in which I have python (I'm learning). I haven't worked with Scrum before.

Nusrat Hossain

- Master's Programme in Embedded System.

While doing the courses I did a little bit of programming in Python but not in depth. I didn’t work with the scrum method before.

Catya Kemppainen

- Master's Programme in Computer and Information Engineering

I have used Python a little before but are not very experienced with it. I have worked with Scrum during multiple projects before.

Anto Juko

- Master's Programme in Computer and Information Engineering

Done some simple Python based programs on Raspberry pi.

Sandeep Sunny

- Master's Programme in Computer Science

I only know basic python and have not used it in any projects before.

Marie Paulsson

- Bachelor's Programme in Computer Science

Done multiple programming courses mostly in C. Some experience with Python. Worked with Scrum in a project before.

Anwar Chowdhury

- Master's Programme in Computer Science
  I have experience working as a researcher and developer. I have developed many Android apps with Java and Kotlin. I have worked with Agile methodology.

Anton Bergåker

- Master's Programme in Computer and Information Engineering.

Done some game development but not in python.

6.1.2. Individual Interests in the Course

Nusrat Hossain - As a student from embedded systems I always did some projects but never knew how the project management was done. Also how to deal with the client requirements, how to do the contract paper, documentation, analysis and so on. This time my interest is to learn all the aspects of project management by collaborating with all of my team members. So I would want to take a role who writes the contract, agreement and also manages the trello board.

Marie P - To get a feeling of how project management is done, to have fun.

Anwar Chowdhury - I am interested in the analysis & documentation.

Srinivasa Sai Akhil Raghupathi - Organizing work has always been one of my keen interests, being a part of an embedded systems course I have just started working on programming like python, I will make sure to contribute all that I'll learn from my python course to the project. I would love to work in one of the roles like Product owner, communication manager, Scrum master where I can collect client requirement specifications,do analyse,research, create documentation and support as backup developer upon requirement.

6.1.3. Roles

- Banker
  - Only one who has access to the team's wallet
- Communication Manager
  - In charge of contact with other groups / client / scrum master

- Client Meeting Facilitator
    - Leads the meeting with the client and goes through our questions
- Secretary
    - Takes notes during meetings with the group / client / scrum master
- Project Manager
    - Manager of the group who makes sure all members do their part and that the things required get done, writes contracts for selling
- Developer
    - Implements the game from the requirements
- Lead Developer
    - Developer who is in charge of the Git and organizing the work
- Documentation
    - Writing the requirements and the deliverables during the course
- Tester
    - Tests the game (by playing it) during various stages to see that it fits the requirements and that there are no bugs

6.1.4. Role list

Nusrat Hossain
- Project Manager, Documentation

Carolina Gripsborn
- Secretary, Documentation

Srinivasa Sai Akhil Raghupathi
- Documentation

Anwar Chowdhury
- Documentation, Tester

Anton Bergåker
- Lead Developer, Client Meeting Facilitator

Marie Paulsson
- Communication Manager, Developer

Anto Juko
- Banker, Developer

Catya Kemppainen
- Developer

Sandeep Sunny
- Developer

6.1.5. Tools

- Slack
    - Communication outside of meetings, questions and planning meetings

- GitHub
    - Storage and review of code for the project between group members
- Zoom
    - For meetings / group work
- Trello
    - Creating and keeping track of tasks during the project
- Google doc
    - Documentation during meetings

6.1.6. Rules / Organization

- We have a chairman/facilitator during meetings who goes through the points of the meeting and is in charge of moderating discussions
- One person takes notes on what was said during the meeting in a shared document (for someone who couldn't participate on the meeting)
- We collectively vote on what our Uppsalines are to be used for in the group
- Only the one who is responsible for our Uppsalines can utilize them according to group decisions.
- Always communicate with the group if you are having trouble or are stuck
- For those who miss project meetings (without logical reason or prior notice), repercussions are discussed within the group
- Everyone must participate in planned activities, otherwise see point above

6.1.7. In the Case of a Free-rider

If we discover a free-rider in the group, the members will talk to them first and ask why they are in this situation. If it's something that can be solved within the group and the free rider is later able to contribute satisfactorily to the group they will be given a second chance. If not, after discussion within the group we might have to contact a teacher to solve the issue.

6.1.8. When / how do we work

- One group meeting per week on mondays at 8.00 where we divide the work
- Group meetings through zoom, individual pairs / smaller groups decide themselves if in person/zoom
- Divide into smaller teams doing different aspects of the project
- Help each other when stuck or having problems

6.1.9. Initial Tasks

- Make requirements according to client's wishes
- Meet with scrum master
- Start writing Deliverable D2 according to client requirement.

6.2. Examples of Meeting Notes

6.2.1. Meeting with Group C

1. Introduce ourselves
    - Who we are
    - What we have done; a user-friendly game platform for the UU-game. It is simple, clean, intuitive and reliable. The game platform works 100%, just some small improvements requested by our client and some addons to implement and documentation.
2. Start demonstrating our component
    - Show the main menu
    - Play the game
    - Show what moves are allowed / not allowed etc.
    - Maybe say what some of our addons/improvements will be (ask for player name, show available adjacent positions etc.)
3. Answer their questions
4. They show their component
5. Ask questions to them
    - What kind of file do you use for the AI?
        i. Json file
    - Will the player only send the AI its move or will the whole game state (updated with player's move)?
        i. The updated board
    - Will the AI generate a completely new file or overwrite the current one?
        i. It will update/overwrite the current file.
    -
6. Talk about the deal
    - Both components will cost the same
    - 2-3 developers from each group will help the other group integrating the components together

Questions for our group about the meeting:

- How many difficulty levels should we have?? I think Amina mentioned 3 but I don't think we asked about it
    - She wants 3 different difficulty levels. 2 is not enough.

- Are they supposed to have 3 different levels since Andreas mentioned 3 levels?

6.2.2. Meeting with client

Attendees: All

- two player board game
- clear, beautiful, simple to use
- play in terminal
- players type to play
- players take turns
- players move their pieces
- several ways to decide if win or lose
- no dices/cards only players that make decisions
- players can move something
- pieces - 12 each (all equal)
- pieces on grid board
- at start no pieces on board, players place
- black and white pieces (one color per player)
- movement: can't go to a position that is occupied
- at one point a player can fly to a position on the board
- can move to all adjacent cells
- win: if one player can't make a move the other wins, too few pieces on the board
- can't jump over other pieces
- player can lose pieces caused by opponent
- have certain configuration of pieces on your board to take out opponents piece
- position is not important
- certain amount (3) of pieces in a row is the key
- certain constraint on what piece is removed
- one move every turn, move or place
- all pieces on the board before you can start moving them
- no passing
- black one starts
- piece that is removed is gone forever
- you can remove piece that is not in a 3-row
- loose if <3 pieces left
- Intersections are the positions a piece can take
- opponent piece removed instantly when 3s are placed
- open game in terminal by writing

From the client meeting, we covered 70% of the game, what is missing are mostly rules.

6.2.3. Monday Meeting 28/9-20

Attendees: All

Agenda:

1. Team meeting with ameena(client),show her the platform if possible.
2. Fix the trello(Check by everyone if anything is missing)
3. Meet the deadline(project)
4. Attend the activities to get more uppsalines(must)
5. Make an Integration plan(is it possible to do another component?If yes then which one?)

Developers Plan: After meeting with ameena they will plan for the next task also decide if it is possible to do another component or not

Documentation Team Plan: Start analysis of example contract agreement paper , do some market research(how)?

6.3. Scrum retrospectives

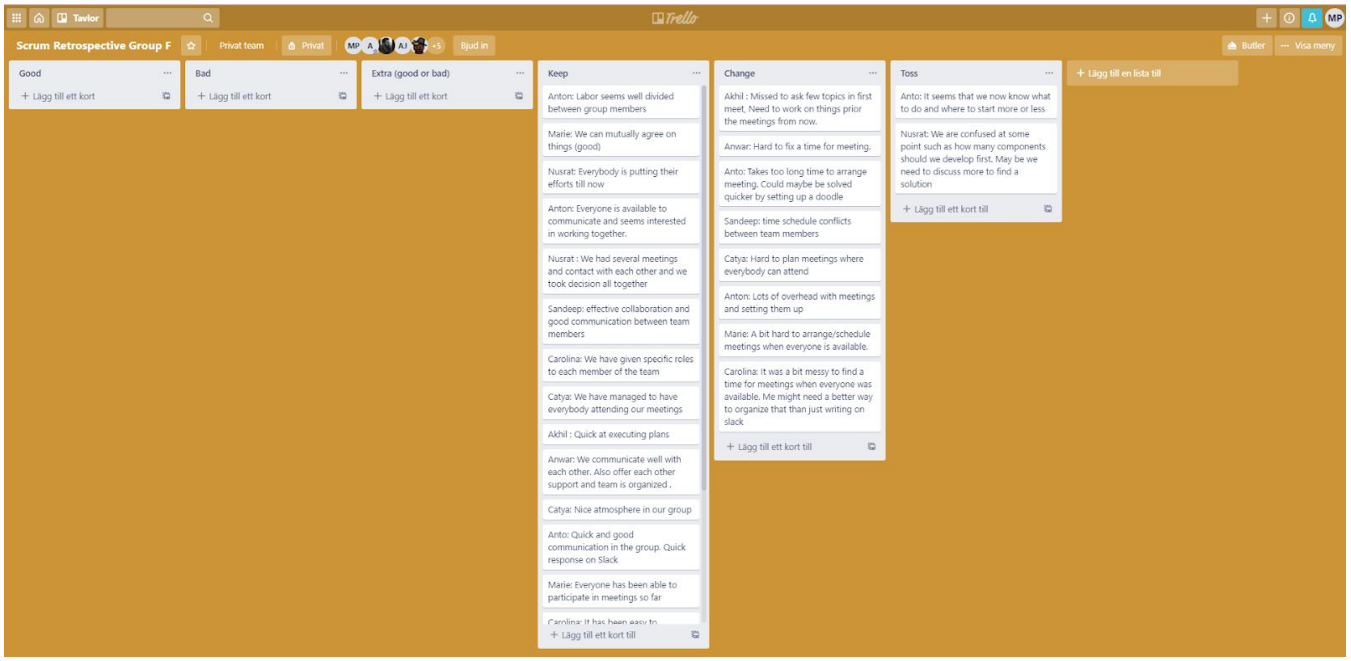

Figure 5: Scrum retrospective 11/9

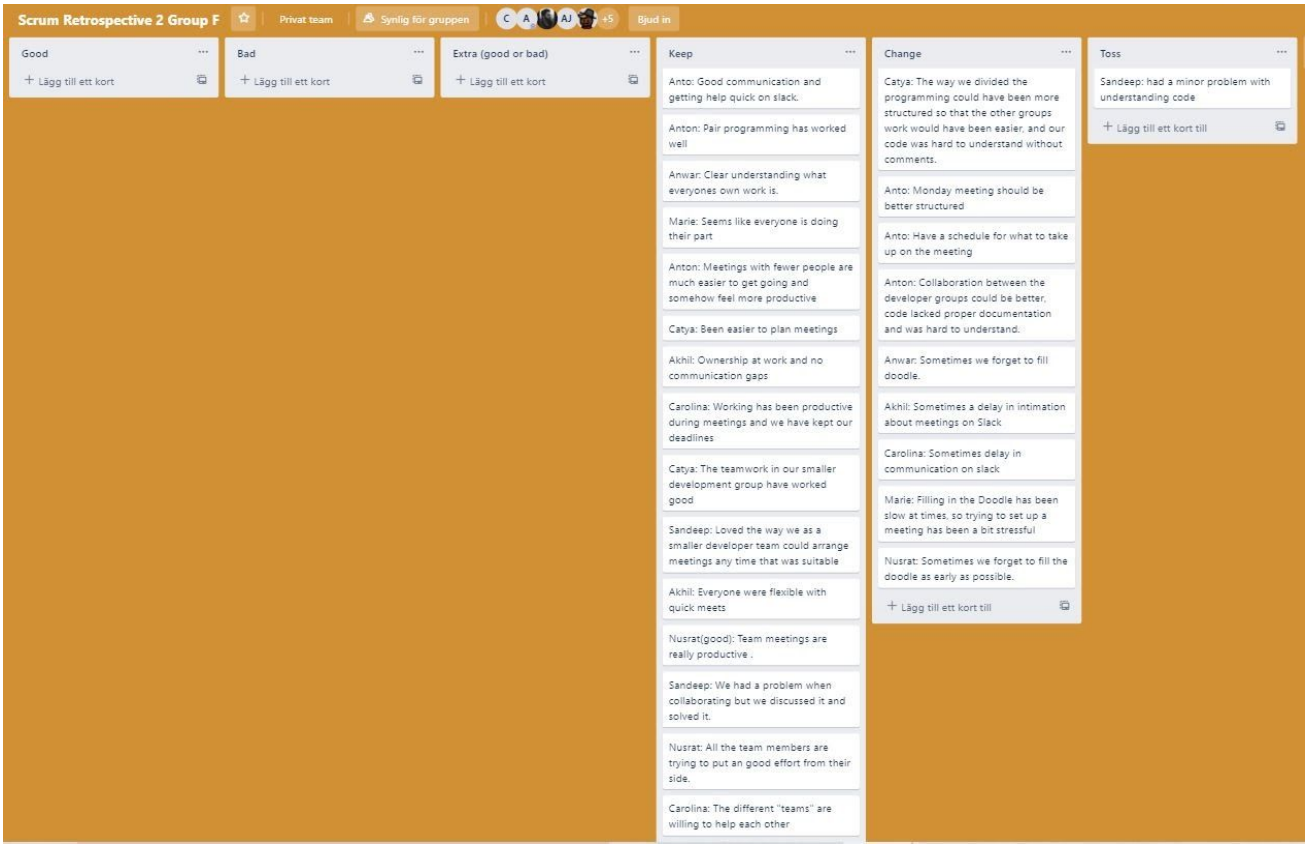

Figure 6: Scrum retrospective 25/9

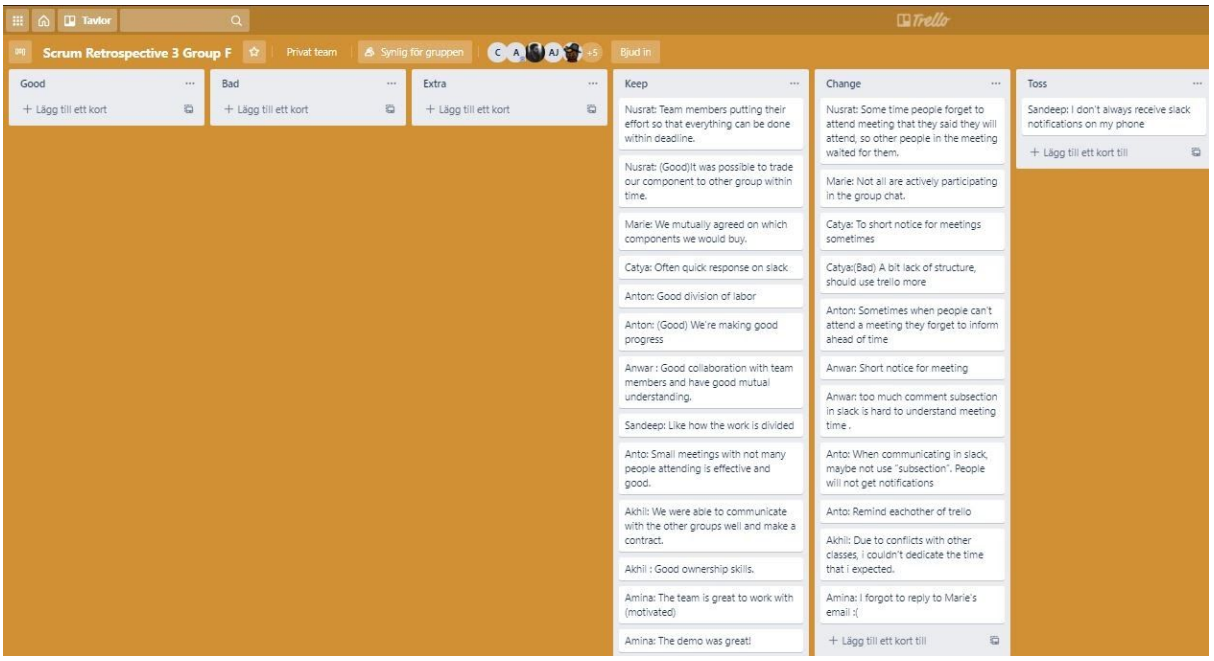

Figure 7: Scrum retrospective 9/10

6.4. Link of Github repository

https://github.com/AntonBergaker/SPM-Group-F

6.5. References

D3 of group F, C & I
D5 of Group F
Project Details, Course page