# 3rd Semester Project

Dinnergeddon – System Development Report


Stefan Nikolaev Borisov
Linda Augustina Carolus Fuchs
Alexander Ignácz
Stefan Jõemägi
Dimitar Bogomilov Pilyakov
Nikola Anastasov Velichkov

UCN – COMPUTER SCIENCES AP DEGREE
DMAI0917 – PROJECTGROUP 7

This page is intentionally left blank.

# University College of Northern Denmark

Computer Sciences Academy Profession Degree Programme

**Class:**
DMAI0917– Project group 7

**Title:**
3rd Semester Project – Dinnergeddon
System Development Report

**Project participants:**
Stefan Nikolaev Borisov
Linda Augustina Carolus Fuchs
Alexander Ignácz
Stefan Jõemägi
Dimitar Bogomilov Pilyakov
Nikola Anastasov Velichkov

**Supervisor:**
Dimitrios Kondylis

## Abstract

Creating working software is only a small part of software development. A much greater part is the choice of proper software development methodologies, the pros and cons of the different types and the decision-making process. In equal degrees, it is important to be able to ensure a high-quality standard for the final product. Which tools can be used and what are the consequences of developing in this way?

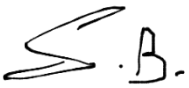
The use of agile methodologies allows for greater flexibility within the system development process and gives the team the chance to change parts of the system as seen fit all with a better implementation in mind. Though one has to be careful that quality does not suffer in the pursuit of implementing features.

This project revolves around the use of the Scrum methodology and the delivery of working software at the end of each sprint for a total of 5 sprints to conclude the project. This also includes the controversial sprint 0.

**Submission date**: 17-12-2018

**Number of characters including white spaces, excluding cover, Table of Contents, tables and appendices**: 41,696

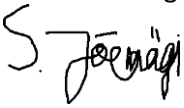
Stefan N. Borisov



Linda A.C. Fuchs

Alexander Ignácz

Stefan Jõemägi



Dimitar B. Pilyakov

Nikola A. Velichkov

*This report has been written as part of the 3rd semester project for the Computer Sciences course at UCN, University College Nordjylland. The main objective of this project according to* (University College Nordjylland, 2014, pp. 7-8) *is to "master more sophisticated elements in the computer science profession and realize distributed software systems; and contribute to the selection and use of technology in the context of system development and programming of distributed IT systems as well as give the students thorough knowledge of aspects of technology." and "make new and further developments and integration of distributed IT systems on a systematic basis using situational modern system development methods and techniques" within the timeframe that has been given by the institution.*

# Preface

We would like to thank **Dimitrios Kondylis** for his continuing support during this project. The project was a challenging, and we set the bar high right from the start. Dimitrios' support and feedback were invaluable to keep the project on track and ensure that we had a healthy view on the reality and achievability of this project.

We would also like to thank **Michael H. Andersen** for his support during the project, giving use some good pointers about what to look out for with our web application and the concurrency issue.

Lastly, we would also like to thank **Simon Schmidt-Jakobsen** for his help with overcoming some of the trickier obstacles we encountered while programming our project.

Thank you,

Stefan B., Linda, Alexander, Stefan J., Dimitar and Nikola.
17-12-2018

This page is intentionally left blank.

# Contents

This page is intentionally left blank.

# Introduction

This report has been written as part of the System Development section of the 3rd Semester project for Computer Sciences AP Degree at UCN Aalborg. The goal for this project is to successfully create software in the C# language which demonstrates the student's ability to create client-server architecture within a piece of software which also solves a concurrency problem. This report shows the steps taken to build working software from a system development perspective, which also takes security into consideration (Nordjylland, University College, 2018).

The rest of this introduction will detail the problem area and -statement in more detail, go into how data is collected and processed, will detail the structure of the rest of the report and finally some logistical information regarding version control and databases used. The team working on this project has agreed to sign a group contract. A copy of this can be found in Appendix 2. This copy is not signed, however by signing this report all group members have agreed to this contract.

## *Problem Area*

In order to get a clear view of the problem, and in order to get some experience dealing with a (fictional) company and (fictional) external product owner, the problem area and problem statement (outlined in the next section) were written from the perspective of Best Design Group (*BDG*), who has been contacted by Worst Production Company (*WPC*) with a request for a retro game. '*WPC'* wants a demo of a game that shows a concept of zombies going rogue in a dinner show so that they can see how their customers would react to it.

'*WPC'* has requested '*BDG'* to create a web application to go with the game so that players could sign-up for a dinner show specifically with their friends. Furthermore, the company can only afford one server.

Due to the '*WPC*'s' limited resources, storage space and server architecture will have to be taken into consideration. As there would be multiple connections to the server, concurrency would have to be accounted for. The company wishes to use a website so that users will be able to download their game, register and reserve spaces for the game online.

This report will delve into the system development process of making the software and which decisions were made going forward in the different sprints.

## *Problem Statement*

Now that the problem area has been clearly defined it is vital to write a problem statement which covers all aspects of the project. The main question which needs to be answered is as follows:

**"How does '*BDG'* create software which is built using C# (.NET framework) and incorporates a client-server architecture with WCF within five Scrum sprints?"**

The programming and technologies report which was written alongside this report will go further into the sub questions which focus on the programming and technologies aspect of this project. In this report the emphasis is put on answering this main question from a system development point of view. For a quick answer, please refer to Chapter 5 Conclusion.

The full problem statement can be found in Appendix 3.

### *Empirical Data Collection*

During the construction of this project, external resources, like books, electronic publications and websites were accessed to gather all information necessary to come to a successful fruition of the project. To ensure these all live up to the high-quality standard which is expected from a higher educational institute, only information which has been gathered through empirical research has been used.

### *Structure of the report*

The main content of this report starts in Chapter 1 with a more detailed look at system development and the different methodologies which can be used for software development with emphasis put on the methodology used for this project, and why, including Boehm's method selection and a risk analysis.

Chapter 2 goes into quality criteria and assurance for the project, including a paragraph dedicated to test-driven development

Chapter 3 takes a closer look at the planning and architecture used in the project.

After the project setup has been discussed it is time to take a closer look at the actual system development during the project. Chapter 4 is split up in 5 different sections, one paragraph for each of the 5 sprints used for this project. It will highlight the difficulties and challenges faced during each sprint and the decisions made to overcome them and ensure better progress in the next sprint.

Closing the report are chapter 5 and 6 which will first discuss the conclusion of the system development process and finally the evaluation which covers the project, as well as group members giving feedback to one another.

Finally, any relevant appendixes which will be referred to in various places the report.

### *Vision for the project*

To create great software solutions to better everyday life for everyone!

### *Logistical Information*

Database: dinnergeddon.database.windows.net

GitHub repository: https://github.com/best-design-group/Dinnergeddon/commits/master

Version commit number: 390

# 1. System Development introduction.

This chapter will focus on system development methodologies, what they are and how they are used. By the end of this chapter it will become clear why the choice was made to use an agile development methodology, in this case Scrum and XP, and what the positive and negative aspects are of each in this project.

## 1.1. Methodologies

"A software development methodology is a way of managing a software development project." (Young, David C.; Computer Sciences Corporation, 2013, p. 1). There are many different methodologies available and each has their own purpose, from very prescriptive Waterfall and Rational Unified Process (RUP), to very loose and adaptive Kanban, and every aspect in between, as is visualized in Figure 1

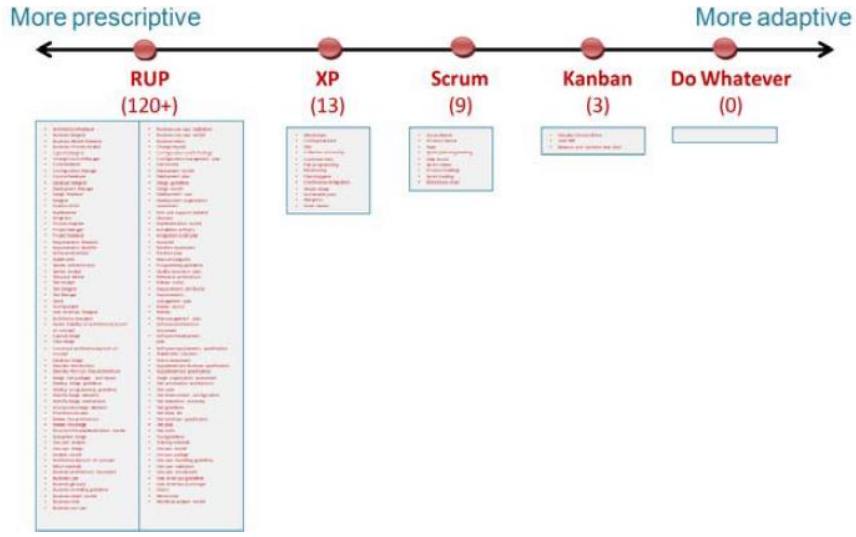



*Figure 1: System Development Methodologies*

### 1.1.1. Plan Driven Development

A few examples of some of the more well-known types of plan-driven development methodologies.

#### *Waterfall*

A good example of a strict plan driven approach is the, now almost outdated, Waterfall method. Waterfall incorporates a lengthy planning phase which often takes up more than half of the project time. Once development starts it cascades, like a waterfall, from part A, to B, to C, without going back to make changes. All requirements are set from the start and every angle is explored. Waterfall is used mainly in situations where the software will be used in situations where any mistakes in the software can have dramatic consequences, including injury and loss of life. Think for example about software for hospital equipment, not a very agile approach and not suitable for the project as adaptability is required.

#### *Rational Unified Process*

The Rational Unified Process has similarities to waterfall, in that is has fixed phases of inception, elaboration, construction and transition, but unlike waterfall, RUP is an interactive process. Every iteration serves as an opportunity to further change and expand the software in an iterative way. RUP relies heavily on the use of artifacts, with a list of over 70 artifacts used, the difference is clear in Figure 1 above. When compared to XP and Scrum, RUP has an attitude of "lets create the artifact just in case we need it" rather than, lets only create it when we need it, like in Scrum and XP.

### 1.1.2. Agile

On the other side of the methodologies' spectrum is a more adaptive and agile approach, and these are also the methodologies used for this project. Scrum is more agile than eXtreme Programming (XP), but both are still much more agile and adaptable than RUP and waterfall (Kniberg & Skarin, 2010, pp. 9-10).

#### *EXtreme Programming (X3M P)*

Extreme programming does not have as many artifacts as (R)UP, but heavily relies on set values and practices. If one of these values falls away, the whole methodology fails. Below a brief explanation of the values and practices and examples of how these were implemented in the project.

**4 Values:**

Communication – The value of communication is critical for the success of the project as it implements one of the core aspects of agile development, constant communication within the team and with the external product owner in order to improve on the project in an adaptive way.

In the project there was a new product owner every sprint. This ensured that each team member had the ability to both practice as product owner and get a better understanding of the product in general. Having the product owner change also meant that the emphasis on the most important parts to implement first changed from sprint to sprint, as they would in a real-life situation.

Great emphasis was also put on having the team together to work on the project on most days, as it is a lot easier to simply tap someone on the shoulder to ask a question, rather than having to pick up the phone or write something over online communication.

Simplicity – In XP it is critical to keep code as simple as possible. Simple code makes for easily readable code and code which is easy to adjust. The whole concept of agile development is the ability to change, so it goes without saying that simplicity goes hand in hand with this. Simplicity also has the benefit of ensuring low coupling.

In the project this same standard has been implemented and it had the positive effect of being able to easily change the code and refactor if necessary.

Feedback – The value of feedback doesn't only rely on feedback from product owner, rather from feedback from all levels on all levels. What that means is that feedback received from an early alpha test will be taken into consideration as well as feedback from the product owner and other stakeholders. This will then be reflected in the planning of the next iteration.

In the project there was constant feedback in the form of asking for assistance from colleagues as well as feedback from teachers and supervisor. Of course, the product owner during this iteration.

Courage – The courage value refers to the courage to experiment with the code and make changes and even reject things as time goes by without formal process. So that parts of the system can be added or removed as seen fit without the product owner being

During the project this was used repeatedly, which in the end resulted in several code refactoring sessions, however it led to a working version at the end of each scrum sprint, and a better piece of software in general.

**12 Practices**

In XP there are 12 practices in total, which are displayed in Figure 2 on the right (www.XProgramming.com, n.d.). All 12 values have been implemented in the project. A few of these will be highlighted below.

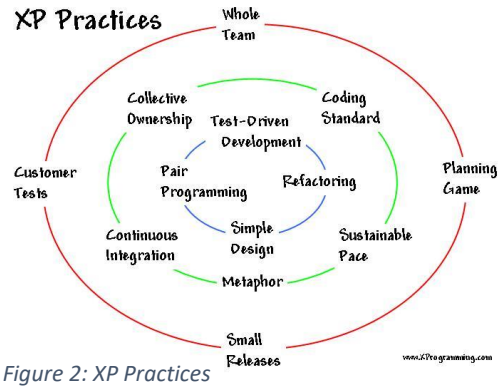


*Figure 2: XP Practices*

Collective Ownership – The entire team is owner of every part of code in the entire project, thus able to edit anywhere as needed. This also means the entire team is responsible of every piece of code written.

Small Releases – Small releases, with a working product at the end of each 1-week sprint, means that the product owner and stakeholders have software that can be tested (also customer tests practice), and they can give feedback on, on a regular basis. Which incorporates both feedback and communication values.

Coding Standards – Coding standards were put in place to ensure that every team member would write code in the same way, think for example about the previously mentioned simplicity value (also simple design practice), which enables high cohesion and low coupling.

Pair programming, Test-Driven Development and Refactoring are all values which have been applied during development, from working in pairs with more skilled and least skilled members together, writing tests for the most critical parts of the system and refactoring code as new parts get implemented.

The use of Extreme programming has great benefits, as it gives the team a set of values and principles to hang on to during programming. These values and practices are all

### *Scrum*

Scrum is a project management methodology rooted in efficiency and flexibility. As mentioned in the official description of scrum by (Schwaber & Sutherland, 2018): “Scrum (n): A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.” In contrast to less dynamic development approaches, scrum lacks a high number of hard-set roles and artifacts. It has 3 roles, namely: Product Owner, Scrum Master and the Team. Scrum also has only 4 types of meetings that happen at specific points in the development process: Sprint Planning, Sprint Review, Scrum Meeting, Sprint Retrospective. Each of these performs an individual function of ensuring future work will be productive and without fault. Scrum also only has 3 artifacts: Product Backlog, Sprint Backlog, Burndown Chart. These keep things organized and accessible.

Scrum sprints are intended to be efficient and dynamic. This is achieved by the Sprint Backlog artifact and the burndown chart for the Sprint. A Scrum team’s velocity is how many tasks they can undertake from the Product Backlog into the Sprint Backlog. It is determined by the total number of story points each of the tasks is rated at. This velocity is set as the beginning of the burndown chart on position 0. After this a predicted velocity is drawn from that start to the last day, where the remaining tasks should be completed, and the points exhausted. At the end of every Sprint day the Scrum Master updates the board to match the actual progress made. This way it is clear how much work is being done and easy to descope and remove less important tasks or add more tasks to avoid slacking. The product owner is the one who oversees the Product Backlog, he adds and removes user stories and changes the priority of these as he, and the stakeholders, see fit.

This way of working offers a highly flexible and agile work environment both for the Product owner and the team alike. Changes are easily added or removed, and the team has a great degree of freedom regarding the how they proceed with the implementation of the selected user-stories.

The downside of this approach is that this method will only works immediately from the start of a project with an already established team. A new team or an inexperienced team will have to take some time to get used to each other and constant changing environment that Scrum offers.

#### *Kanban*

The last methodology to be described in this report is Kanban. When compared to scrum, Kanban is an even more free and agile approach to software development and has less required roles, rituals and artifacts. In Kanban there are only a few rules. The use of a Kanban, the physical board which keeps track of development, and the Kanban cards which move around on the board.

This is not a good method to be used for an inexperienced group however, and the use of additional artifacts is often welcomed. For example, the use of backlogs and graphs in order to keep track of features and progress in the project, if only to be able to report back to an external product owner. As is clear from this example, if is easy to fall back to a more Scrum like methodology.

## 1.2. Evaluating the methodology selection using Boehm's factors

To determine whether the project's development is on either the agile or disciplined side, Boehm's five critical factors can be used. These five major decision factors are size, criticality, dynamism, personnel and culture. For this evaluation process, (Boehm & Turner, 2003) was extensively consulted.

By rating the project along each of the five axes it is possible to evaluate its position on the agile-disciplined scale. If all the ratings are at the periphery, a disciplined, plan driven approach is recommended. However, an agile methodology is suitable if they are located near the centre.

After summarizing the factors, a diagram was created. As shown in Figure 3 on the right page, the Size, Criticality and Culture factors are close to the centre which points to a more agile approach, but according to the Personnel and Dynamism axes a plan-driven approach would be more suitable.

Since the percentage of 1B people working on the project is above 20%, a continuous presence of Level 2&3 would be required in an agile environment, while using a disciplined methodology could accommodate some Level 1B people.

As far as Dynamism is concerned, a huge percentage of requirements are not expected to change, seeing as these are directly linked to the requirements set by the educational institute. That is one of the features of a plan-driven methodology.

The culture axes show a preference to the 'chaotic' side, due to the fact that the team is enthusiastic, and feel empowered by the prospect of learning about their own ambitions; Game Development.

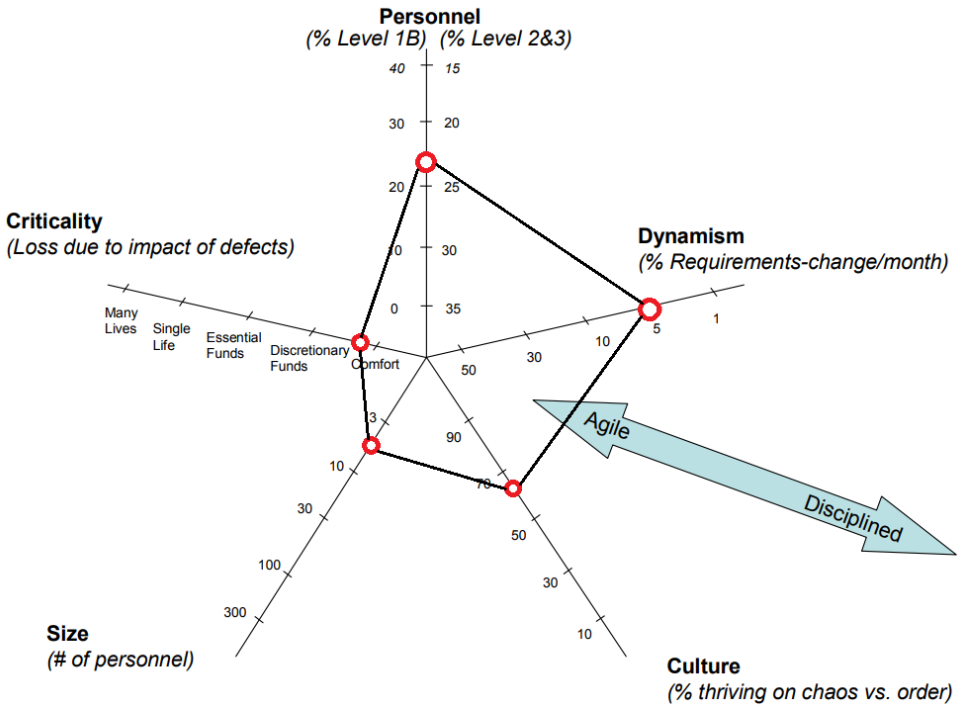



*Figure 3: Dimensions Affecting Method Selection Diagram*

Despite the two factors of personnel and dynamism suggesting a disciplined methodology, an agile approach is more suitable for this project according to the size, criticality and culture factors.

## 1.3. Risk Analysis

A risk is something that influences the project plan or the schedule in a negative way. By analysing the risks, unnecessary delays can be avoided concerning the project. The steps for creating a risk analysis are the following: identify potential risks, assign probability, evaluate consequences and find a way to handle them.

The following table (Table 1: Risk Analysis table) displays the risk for the project. Each risk has a probability and severity associated with them on a scale of 1-5 and a total threat score which is calculated by multiplying the probability and severity points. After calculating their probability and severity a short solution was figured out which can either prevent a risk from happening or handle its occurrence.

*Table 1: Risk Analysis table*

| Risk | Probability | Severity | Total | Handling |
|---|---|---|---|---|
| Communication issues | 3 | 5 | 15 | Practicing conversational mindfulness and having someone oversee group conversations and act as a mediator. |
| Health problems | 3 | 4 | 12 | Tasks are redistributed, or burndown chart is updated. |
| Limited Sprint length | 3 | 4 | 12 | Overestimating user stories, not allowing slacking and sticking to most-critical-first when developing. |

| | | | | |
|---|---|---|---|---|
| Bad estimation | 4 | 2 | 8 | Reflecting on previous sprints and why the estimation was wrong. |
| Difficulty adapting to Scrum | 2 | 4 | 8 | Ensuring everyone understands and sticks to the principles of Scrum. |
| Changes to the project | 2 | 3 | 6 | Ensuring the project is built with low coupling. |
| Unknown technologies | 2 | 3 | 6 | Research from all available resources in order to form a better understanding. |
| Version control problems | 3 | 1 | 3 | Going back to previous stable versions, having separate & isolated development branches. |
| Hardware malfunction | 2 | 1 | 2 | Ensuring a backup of vital resources is kept on a remote server. |

# 2. Quality Assurance and Quality Control

This chapter will elaborate on what the quality criteria for this project are, and how this quality can be assured. Quality can best be described as follows: “Quality is one or several persons assessment on whether their expectations to and experience of a product is the same.” (Kondylis, 2018, p. 8)

In order to keep good quality of the product multiple quality assurance methods were used. One of the most important ones is test driven development. As described above in the Dinnergeddon project, the critical components were tested using TDD. This ensured that even after any refactoring, the functionality of the components would remain the same.

Another method used during the development process was to constantly communicate with the product owner, asking questions like “How do you like the user interface so far?”, “Does it lack any functionality?”. This ensured that the product owner is happy with the progress so far and if there are any parts of the system that they are unhappy with, they can be refactored or removed.

Pair programming also played a big part in the quality assurance as it allows the programmers to keep each other in track with any mistakes that are made during development.

## 2.1. Test Driven Development

Test Driven Development (TDD) is the act of creating tests before code, so that they fail and then refactoring code to pass the tests. This method of development ensures that functional code will be written in a disciplined and clean way (Ambler, 2018).
For the purposes of Dinnergeddon’s development, TDD was applied to the **critical components** of the project. This was to ensure those components would be implemented smoothly and with a minimal to non-existent amount of slowdown to the development process. This also saves costs of future need to repair or modify software and makes it easily-maintainable. Some of the principals implemented for the project were focused on creating stubs/mocks of data, classes and methods. These all helped contribute to isolating code to the smallest piece of functionality that could be tested with a plethora of data to ensure proper functionality (Kondylis, 2018).

# 3. Planning and Architecture

This chapter will cover both the planning of the project as well as the conceptual architecture for the project. The project plan will highlight the goal of the project and a project plan divided into sprints, while the architecture section will cover the client-server architecture behind the Dinnergeddon project.

## 3.1. Project Plan

The overall end goal of the Dinnergeddon project was to produce an online co-operative zombie-shooting game with a client-server architecture and web site. These main components were broken down into smaller and easier-to-process tasks, called user stories and estimated in sprint 0 in preparation for the rest of the sprints. The time frame for this was 5 weeks of 37 working hours each. The process took place under strict deadlines, which had to be accounted for when planning the development process. Below in Table 2 is the project plan, as far as it is possible to plan agile development.

*Table 2: Project Plan*

| Sprint # | Description | Hours |
|---|---|---|
| **Sprint 0** | Getting the team on the same page,<br>Setting up version control,<br>Estimating user-stories | 37 |
| **Sprint 1** | Implementation of the most critical parts of the project | 37 |
| **Sprint 2** | Additional features | 37 |
| **Sprint 3** | Additional features | 37 |
| **Sprint 4** | Additional features | 37 |
| **Future development** | Additional features to be implemented outside the scope of the project | xxx |

## 3.2. Architecture

Once the project plan has been finalised it is possible to plan for the actual architecture of the system. To the right in Figure 4 is a visual representation of the architecture of the server.

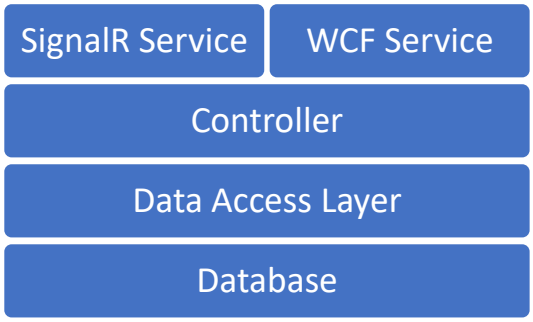



*Figure 4: Server Architecture*

It elaborates how the internal projects are connected to each other, from the database up until the two services which expose functionality to the internet and processes within the same server.

A client-server architecture was used for the Dinnergeddon project. A client-server architecture is such that the server is on one machine, often connected to the internet, that exposes information or functionality about the product. The client, on the other hand, can connect to that server and consume that information or functionality. (Sommerville, 2016, pp. 180-182)

The clients themselves are the web client and the dedicated WPF client. The web client is being handled by an ASP.NET service, which also lives on the same server which allows for using WCF services that are restricted, and sends HTML, CSS and JavaScript files to users. The WPF client connects to the SignalR Service and the WCF Service to consume their functionality from the web.

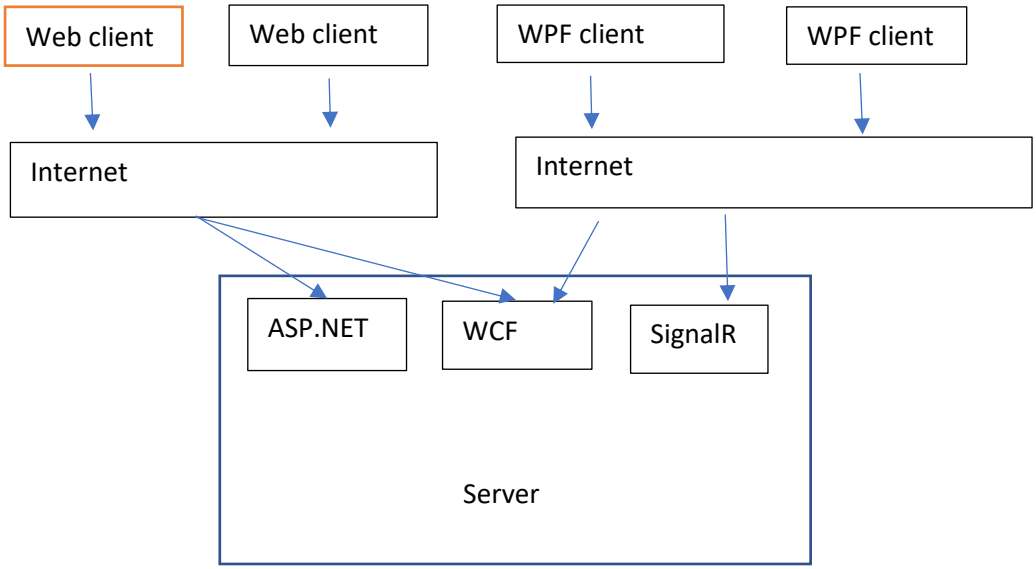



*Figure 5: Client-server architecture with external connections*

In Figure 5 above is a visual representation of how clients connect to the different services on the server. The web clients connect to the ASP.NET server, which in turn uses the internal and external functionalities of the WCF service. The WPF clients, on the other hand, connect to the external functionality of the WCF service and the SignalR server.

# 4. Reflection on the agile methodology in the project

The previous chapters explained everything that happened before the actual start of the project, the sprints. This chapter will detail what happened in terms of system development during the sprints. Each sprint will be mentioned separately in its own paragraph with the most important decisions explained.

## 4.1. Sprint 0

Sprint 0 was an opportunity to plan and start setting up for a smooth start to the project. Since the overall duration of the sprints is less than what is the standard according to (Schwaber & Sutherland, 2018), and because the team working on the project is inexperienced with scrum and have not worked together before, sprint 0 was required in order to ensure a better first sprint and overall performance during the sprints. It was decided that among other things, the Pomodoro Method would be used to work with, since it has been proven to increase productivity for certain tasks and prevent loss of focus on what is currently being worked on. According to (Cirillo, n.d.). Together with this it was also decided that the story point evaluation would be based on Pomodoro blocks (each consisting of 5 individual pomodoro sessions of 25 minutes, with 5-minute breaks in between), which were estimated to be 2 and a half hours. This meant that 1 Pomodoro block was equal to 1 story point. This was done to ensure that the limited time would be used efficiently to accomplish the most work. This sprint was also used to set up all the necessary software (Visual Studio), version control (GitHub), the product plan (), product backlog and estimated the user stories with planning poker.

## 4.2. Sprint 1

### *Backlog*

There was a total of 44 story-points in the sprint backlog at the start of sprint 1.

### *Review*

The sprint started with two days of almost no productivity, even with strict following of the Pomodoro Method. On the third day stand-up meeting the group agreed that the same efficiency technique was hurting productivity and costing precious time. After this revelation the technique was dropped and the burndown chart (shown below in Figure 6) started going down normally. Besides these difficulties there were also inaccuracies in the point estimation for the user stories, which eventually led to a large descoping on day 4, down to 13 story points for the sprint, which were nearly met in order to keep an accurate representation of the team velocity, the choice was made not to remove the stories from the backlog straight away. This way it was possible to better estimate (or so was the idea) how many user stories could be tackled in the next sprint).

Figure 6 below shows the board at the end of sprint 1, with a team velocity of 10

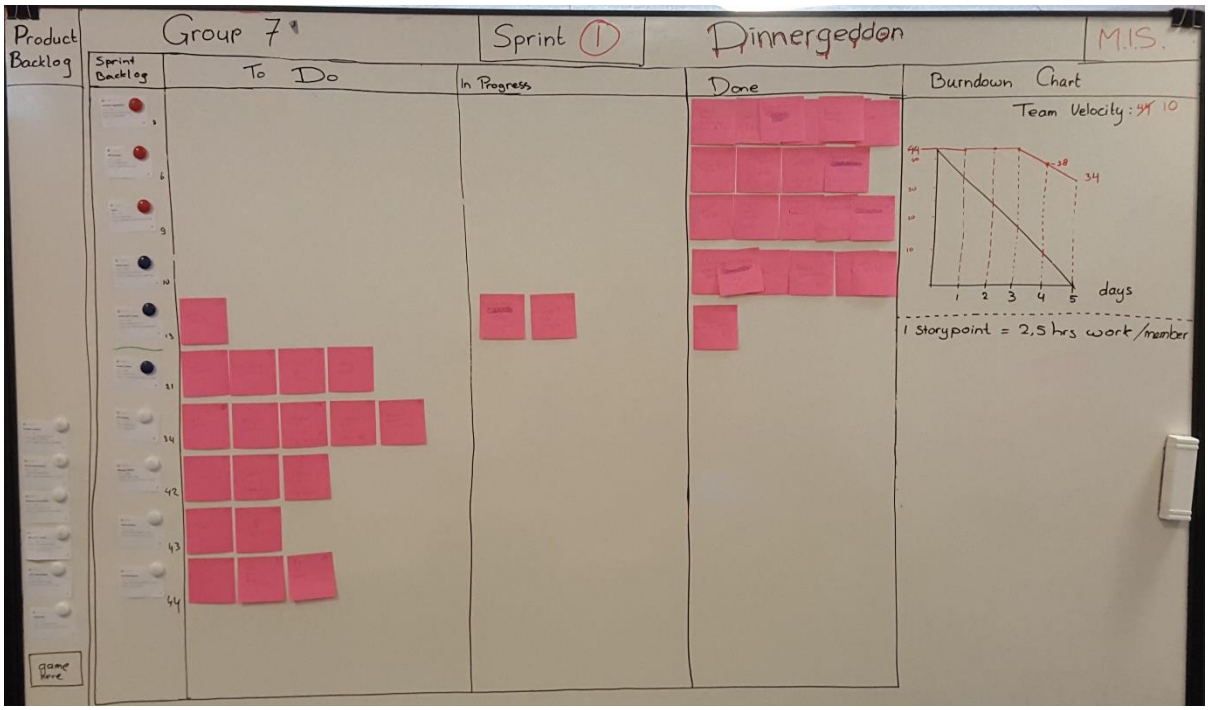



*Figure 6: Board at the end of Sprint 1*

Product Owner: Stefan J.

Scrum Master: Linda

## 4.3. Sprint 2

### *Backlog*

There was a meagre 10 story-points in the sprint backlog at the start of this sprint. It quickly became clear the team could handle more, adding an additional 15 points (the burndown chart shows 13, this is a mistake).

### *Review*

This sprint started out with the opposite issue. The stories in the backlog were tackled quickly, within the first two days. After this more stories were added to the backlog in order to maintain productivity. These were also completed in the remaining two days, giving a better idea of how much work can be handled in a sprint.

Figure 7 below shows the board at the end of sprint 2, with a team velocity of 25. It should be noted that the burndown chart shows 5 days for this sprint, though in reality there were only 4 days, as 1 whole day of this sprint was consumed with sprint reviews and presentations thereof.

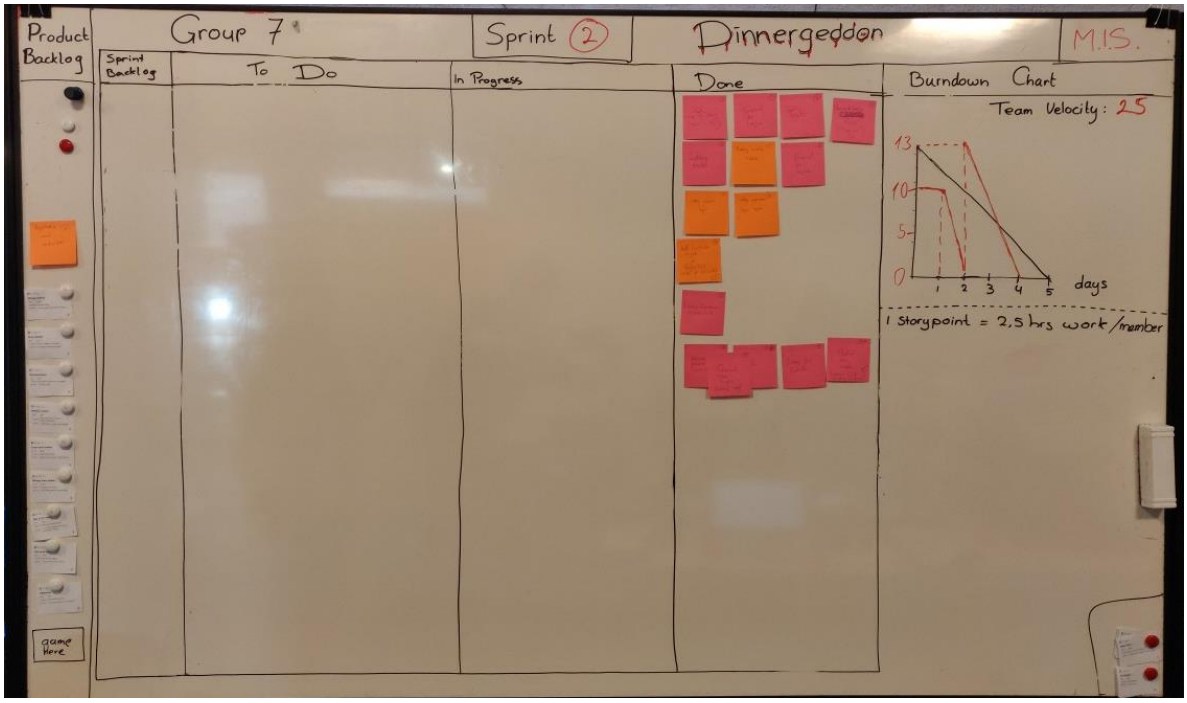



*Figure 7: Board at the end of Sprint 2*

Product Owner: Linda

Scrum Master: Dimitar

## 4.4. Sprint 3

### *Backlog*

There was a total of 25 story-points in the sprint backlog at the start of the sprint. Right from the start the team was proceeding fast so an additional 5 points were added. Again, more fast progress was made, and another 3 points added.

### *Review*

After the hardships during the first two sprints, the planning disasters and inaccurate estimations many lessons were learned. Sprint three was a prime example of that, as it was notably different in many aspects. This could be easily picked in both planning and working aspect. The group's organization, preparation and estimation were much better. The group was however not working at full efficiency due to an ill member. Sprint three started with the addition of some new user stories and refactoring of old ones. These stories were promptly completed close to the burndown estimation. Since a part of the project consisted of a game, the last days of sprint three were used to set up the Unity project and do the required research. The very last day of the sprint was filled with excitement as work on the actual game began smoothly and productively. Unfortunately, the last user stories could not be finished in time thus 5 story-points remained on the board, this also (as mentioned) due to illness of one team-member.

Figure 8 below shows the board at the end of sprint 3, with a team velocity of 28.

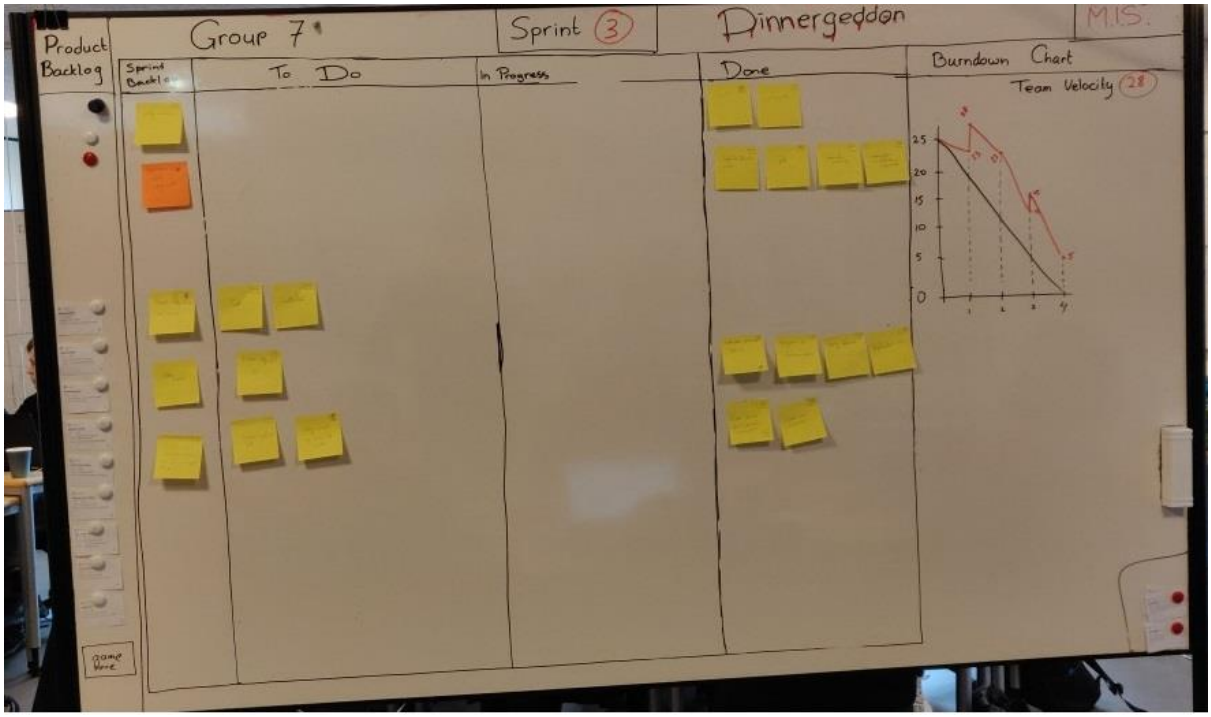



*Figure 8: Board at the end of Sprint 3*

Product Owner: Stefan B.

Scrum Master: Nikola

## 4.5. Sprint 4

### *Backlog*

There was a total of 37 story-points in the sprint backlog at the start of the sprint. All user stories were finished and a new small story (1 point) was added to ensure no time was wasted.

### *Review*

The best sprint. An opportunity was given to work on the more time-consuming user stories and add major functionality to the project, and this opportunity was seized. Although the first three days do not show any points going down, a lot of concurrent tasks were being performed, and on day 4, they reached completion. This created a dip and the addition of one more user story, which all got completed by the last day of the sprint. Incredible progress was made towards a nearly-finished product.

Figure 9 below shows the board at the end of sprint 4, with a team velocity of 38! (the burndown chart shows 37, however this was incorrectly updated, as the additional 1 story-point was not taken into consideration, something to keep an eye on in future projects)

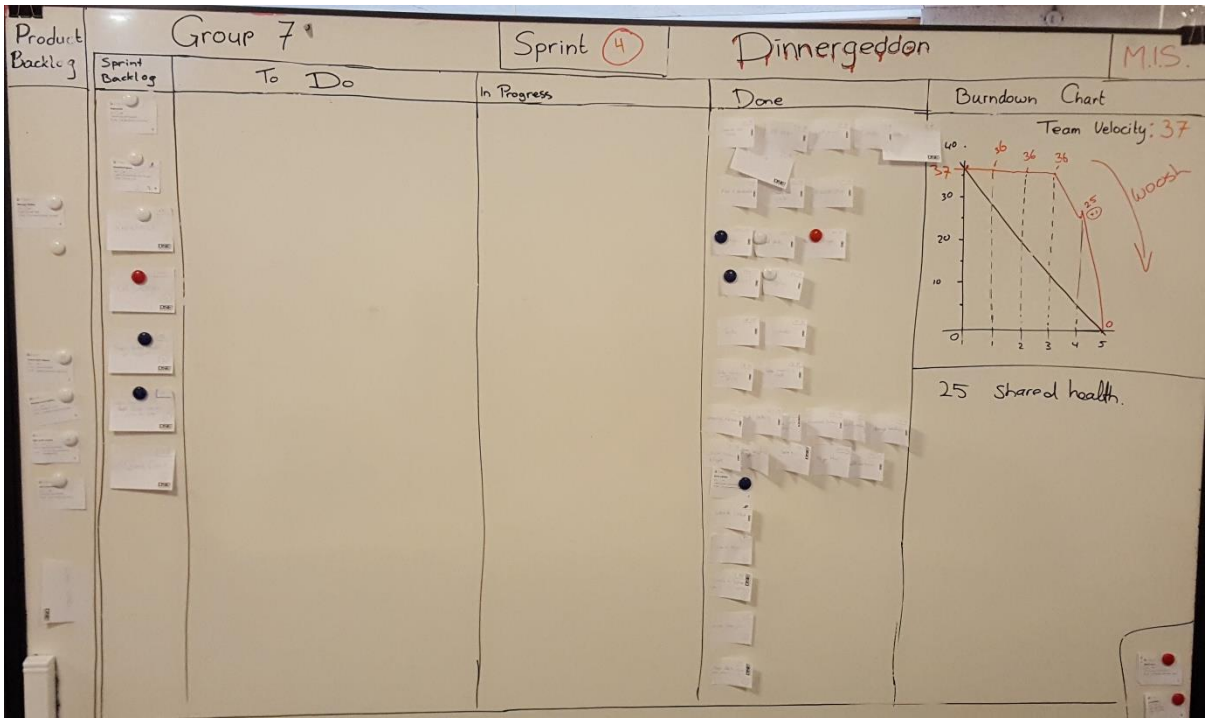



*Figure 9: Board at the end of Sprint 4*

Product Owner: Alex

Scrum Master: Stefan J (& Stefan B in case of absence)

# 5. Conclusion

In order to draw an accurate conclusion, it is vital to first look back to the research question at the beginning of the project, which is as follows;

**"How does '*BDG*' create software which is built using C# (.NET framework) and incorporates a client-server architecture with WCF within five Scrum sprints?"**

### *Discussion*

To answer this, let's first have a brief recap of the project and the sprints. In sprint 0 the team was put on the same page and the infrastructure was set up. Sprint 1 incorporated the most vital aspects of the system, whereas sprint 2 served as a refactor and elaboration on the first sprint.

At the end of sprint two the basic software, built in C# with the .Net framework, has been implemented using a client-server architecture with WCF. The barebones for these requirements have taken 'only' two sprints (+1 for setup). Including the solution to the concurrency issue of having multiple connections to the same server and database.

The remaining two sprints were used to further elaborate on the software and improve upon the already existing product. Test driven development was dropped at this point, as the most critical parts of the system were functioning properly and the implementation of small new features, it slowed down progress more than it increased in quality.

Product owner and scrum master changed throughout the project, to give every group member a chance to gain some experience in this position. The downside of that is that the quality of either of the roles is inconsistent throughout the project.

Another discussion point is the duration of the sprints. The minimum viable length for a sprint according to (Schwaber & Sutherland, 2018) is at least 2 weeks, however due to constraints in the educational institute this was not possible. While it is understandable that there must be constraints in order to fit all the learning outcomes into the time that is allotted for this project, it also gives a wrong representation of the real world and hinders the implementation of certain features, for example working in test drive development on small features. In reality however, a 5-day sprint turns to 4 days due to sprint retrospective and presentations leaving little time to properly do TDD.

### *Conclusion*

The team worked well together, and the use of an agile methodology had its positive effects on a fruitful collaboration and finalization of the project and this same methodology would also be used again in the future for development of a project of the same size and scope. Whoever with the change of having a dedicated product owner who can keep an eye on the requirements and a dedicated scrum master who can keep the team motivated and keep track of the activities.

The quality assurance, in terms of Test-Driven Development, was implemented only in the core aspects of the product. In future projects however, sprint length will have to be longer to assure enough time to work in a TDD way and retain the high-quality standard.

# 6. Evaluation

The following chapter evaluates the project and the individual team members. Each section will look back on the development process taking into consideration both planning, personal preference and decision making. Individual feedback will also be given in the form of reflection on personal goals, skills and knowledge.

## 6.1. Project Evaluation

The project for the semester was very challenging, but also very exciting. As soon as the project started, the group had to overcome the need for better communication, since the two computer science classes were merged together, and half of the group's six members came from one class, while the other half came from the other class. Despite the challenge that having six introverts in a group presented, everyone slowly learned how to effectively communicate with each other. Most of the group's members had the desire to continue their education by specializing in Game Development, however news came around that such a course was not to be held for the next semester. As disappointing as it was, the loss of one opportunity only paved the road for another and prompted a collective decision for developing a game and getting much desired, but educationally unavailable experience within this sphere. For a short period of time, during the start of the project, the excitement transitioned into doubt, as no one had any game programming experience, and it was even hard to grasp how to do things. No one even knew how and where to start. Proper communication, and refusal to give up on this great idea, helped diminish the notion of insecurity, and work slowly started. The lack of knowledge on the topic not only was not detrimental, but proved very valuable, since the time spent on reading and researching brought group members together in learning many new things and acquiring a broader range of skills. Around the middle of the project each member's individual preferences made them de-facto an expert in a different field of programming, which was extremely beneficial for the overall shape of the project, and everyone was quite satisfied with the achieved outcome.

## 6.2. Individual evaluations

This section will go into the individual evaluations of each group member. The feedback given below is feedback from the whole group to the individual person, starting off with a section for the group.

### *The group*

Each group member had to deal with their own (varying degrees of) social anxiety not only towards the other members they were familiar with, but especially towards the ones that were new. Despite this, everyone was focused on achieving the common goal, and everyone was trying to improve the communication within the group. Slowly but surely each one learned how to patiently listen to what the others have to say. Then, after everyone has expressed their opinions, concerns, suggestions and advises, a deliberation would take place as to what the best solution for taking on a problem or planning ahead would be. This helped setting up goals and workflow that everyone would have a clear understanding for. While communication kept improving throughout the project, it was no longer something that was consciously brought up. Even though everyone was engaged in every aspect of the project, personal desire for success and productiveness helped each group member greatly contribute in their own field of expertise and the quality of work was easily noticeable. This not only improved the overall shape of the end-product but pushed everyone one step ahead into personally acquiring new knowledge and skills.

*Stefan Nikolaev Borisov*

Stefan had good ideas throughout the project and his opinions were highly appreciated. However sometimes these good ideas and opinions were overshadowed by his sense of humour, which in turn led to some awkward moments.

Possible points for improvement:
Taking on a leadership role in a project, can have a positive effect on his overall sense of responsibility and improve his productivity by keeping him focussed on the long-term goals.

*Linda Augustina Carolus Fuchs*

Throughout the whole project, Linda proved to be best suited for Scrum Master. Her organizational and administrative skills helped the group be organized and cooperative while also inputting great theoretical knowledge throughout the development process.

Possible points for improvement:
Although Linda was very helpful to everyone in the group, that turned out the be a con when she spent too much time trying to help others, while not focusing on the task at hand. What she should do is devote more time to completing her tasks before helping group mates.

*Alexander Ignácz*

Alex was incredibly positive and doesn't easily get discouraged or allow his morale to fall. He was also productive in the tasks he does, without the need of help from the group. Alex was very adaptable to changes as well.

Possible points for improvement:
He should be more confident in his ideas and opinions. They should be proposed during discussion or conversations because they are valid and educated.

*Stefan Jõemägi*

Stefan was very similar to Alex in that he was easily productive with any task he undertook. His morale was also difficult to contest. He was very adaptable as well.

Possible points for improvement:
Speak up as well, he was the most silent from the group. Should say out his opinion more frequently, not only when being told to do so.

*Dimitar Bogomilov Pilyakov*

One of Dimitar's strongest assets was being capable of ensuring that the team remained focused when working and having discussions.

Possible points for improvement:
However his attention to deadlines and staying focussed was slightly too much sometimes and caused him to 'freak out' over smaller details. This could be improved by having a timely conversation with the group when there is has a problem. In a leadership position Dimitar can be more steadfast of his own opinions and knowledge, as his contributions are greatly valued, though he sometimes does not see this himself.

*Nikola Anastasov Velichkov*

Nikola was among the most productive group members. He was very enthusiastic in taking on any role, including the role of a leader. He was the one to push people to express their opinions.

Possible points for improvement:

Nikola should be less concerned about being pushy and be more confident that his solutions are valid. He should also remain more focused on his tasks and not get distracted by small details.

# Appendix 1. Literature list

Ambler, S. W., 2018. *Introduction to Test Driven Development (TDD).* [Online]
Available at: http://agiledata.org/essays/tdd.html
[Accessed 2018].

Boehm, B. & Turner, R., 2003. *Rebalancing Your Organization's Agility and Discipline,* s.l.: s.n.

Cirillo, F., n.d. *Pomodoro Technique.* [Online]
Available at: https://francescocirillo.com/pages/pomodoro-technique

Kniberg, H. & Skarin, M., 2010. *Kanban and Scrum; making the most of both.* s.l.:C4Media.

Kondylis, D., 2018. *Quality Assurance and Quality Control Presentation.* [Online]
[Accessed October 2018].

Kondylis, D., 2018. *Test Driven Development presentation.* [Online]
[Accessed October 2018].

Nordjylland, University College, 2018. *Project_3 (dmai0719).* [Online]
Available at: https://ucn.instructure.com/courses/12801
[Accessed October 2018].

Schwaber , K. & Sutherland, J., 2018. *Scrum Guide.* [Online]
Available at: https://www.scrumguides.org/scrum-guide.html
[Accessed October 2018].

Sommerville, I., 2016. *Software Engineering.* 10th ed. Harlow: Pearson Education Limited.

University College Nordjylland, 2014. *Curriculum for the Academy Profession Degree Programme in Computer Science. National Section.* [Online]
Available at: https://www.ucn.dk/
[Accessed 5th November 2018].

www.XProgramming.com, n.d. *XP Practices.* [Art].

Young, David C.; Computer Sciences Corporation, 2013. *Software Development Methodologies.* [Online]
Available at:
https://www.asc.edu/sites/default/files/org_sections/HPC/documents/sw_devel_methods.pdf
[Accessed 18th May 2018].

# Appendix 2. 3rd Semester Project Group Contract

Group 7 – Best Design Group

- Deliver a completed and functional product
- Learn to write clean code that adheres to proper code standards and naming conventions, making it easy to follow the three pillars of object-oriented programming and make changes to the program if needed.
- Write down a separate section for these code standards in the programming report, highlighting, among other things, naming in 'Camel Case', C# get and set standards, tab width, and code column size.
- Attractive and usable user interface for both website and the developed application that takes into consideration the 10 heuristics of proper GUI development
- Be present and participate in the stand-up sprint meetings. Be proactive and take initiative.
- Be responsible in group work, in case of absence inform the group and if possible, work from home. The group should be able to show understanding to personal situations.
- Pair programming (2 devs., 1 keyboard) and switching from day to day to learn to write C# and use the .NET framework together.
- Make decisions as a group and keep discussing the pros and cons until we can have a unanimous decision.
- BE ON TIME! this includes being back on time from lunch breaks during class.
- Have some team building exercises during the months.

# Appendix 3. Problem Statement

| **BDG - Dinnergeddon** | |
|---|---|
| **Student names** | Stefan Nikolaev Borisov<br>Linda Augustina Carolus Fuchs<br>Alexander Ignácz<br>Stefan Jõemägi<br>Dimitar Bogomilov Pilyakov<br>Nikola Anastasov Velichkov |
| **Title (initial)** | Dinnergeddon |
| **Subject** | Best Design Group (BDG) has been contacted by Worst Production Company (WPC) with a request for a retro game. WPC wants a demo of a game that shows a concept of zombies going rogue in a dinner show so that they can see how their customers would react to it.<br>WPC has requested BDG to create a web application to go with the game so that players could sign-up for a dinner show specifically with their friends. Furthermore, the company can only afford one server. |
| **Problem / Problem area** | Due to the company's limited resources, storage space and server architecture will have to be taken into consideration. As there would be multiple connections to the server, concurrency would have to be accounted for. The company wishes to use a website so that users will be able to download their game, register and reserve spaces for the game online. |
| **Problem statement** | • How does BDG create software which is built using C# (.NET framework) and incorporates a client-server architecture with WCF within five Scrum sprints?<br>  ○ How is a seamless user experience using WPF for a dedicated client achieved?<br>  ○ How is a seamless user experience using HTML, CSS and JavaScript for a web application created?<br>  ○ How are possible concurrency issues which can occur in a multiple user environment solved? |
| **Method / procedure** | During the project, Scrum and XP will be used as the main development methodology. For the back-end, C#'s WCF service has been chosen as a means of connecting to the web application and the game itself. The web application is going to be developed using various web technologies consisting of HTML, CSS and JavaScript primarily. The user interface of the dedicated client will be developed using WPF. The database will be created using Microsoft SQL. |