Evaluation of Iterative Development

KTH Royal Institute of Technology

Modern Software Development

May 15, 2023

Kenan Dizdarević
CINTE21
kenandi@kth.se

Abud David Zoughbi
CINTE21
zoughbi@kth.se

Abstract

This report presents the results and analysis of a project conducted using Scrum, an agile development methodology. The project spanned over four weeks, with each week being an iteration, a sprint. All data collected includes information on task management, specifically the amount, adaptability, and project progress in terms of satisfaction levels. The team's management of time and resources was measured through the number of tasks initiated, completed, added, and removed, as well as the completion of whole stories and story points. Insights on each iteration's success, highlighting achievements and areas for improvement were provided by the product owner. The report emphasizes the team's learning curve, resource management challenges, and the overall progress made during the four weeks. The findings in this report contribute to understanding the effectiveness of Scrum, and especially the practice of *Iterative development*, in a group full of inexperienced developers and provide valuable insights for future projects.

Keywords

Increment, adaptive, effort estimation, time and resource management and software development.

Contents

1 Introduction

Conducting projects is something that always requires good coordination, independent of how many members will participate in the project. Managing resources and time is something which is crucial to conduct a good and healthy project. This is often viewed as something which is hard to do accurately. To help project teams, development methodologies have been developed and employed. The methodologies provide frameworks and guidelines on how to effectively manage and conduct projects. They utilize different principles, practices, and processes to improve productivity. In the end, the main goal for every team is to make a functional increment in the product they are creating.

Iterative development is one of the key principles of the agile development method Scrum and it is characterized by its cyclical and incremental nature. The main goal of iterative development in the software engineering industry is to, early on in the project, deliver value. Iterative development is constructed so that teams can easily adapt to changing requirements. Another focus point of iterative development is collaboration between group members, a key principle of this is transparency during the whole project. Communication between the development team, product owner, and customers is promoted. Feedback from product owners and customers is something that happens frequently.

During our project course at KTH Royal Institute of Technology, we adapted the principles of iterative development. We utilized the framework Scrum, with its guidelines. Iterative development is built upon different factors such as managing the available resources, effort estimation, and continuous feedback. Our team encountered problems with time and resource management, adaptability, and the definition of functional increment. We had no insight into how effective iterative development actually is. Therefore we decided to conduct research on the effectiveness of iterative development by comparing our gathered data from the project with other researchers' findings and reports.

In summary, this paper will give the reader an introduction to the software development methods which are used. An insight into iterative development will be given to the reader in Section 2. Section 3 will outline the research method used when conducting our research. The data which we gathered during our project will be showcased in Section 4. Section 5 will present the analysis of our research and compare our results to other findings which we have made. In the final part, Section 6 we will summarize the key findings and insights from the study. The references which we have used can be found in Section 7.

2 Background

2.1 Software development methods

Software development methods refer to processes and techniques used to develop, design and maintain software. There are two main categories of software development methods: traditional and agile. The choice of development method depends on multiple factors, including project requirements, level of complexity, team size, and many more.

2.1.1 Traditional development methods

When software requirements are established and stable traditional software development methods can be used. These types of methods are often referred to as heavyweight approaches because they include heavy processes such as detailed planning and documentation, building and testing according to the written requirements, and so forth [1]. This results in a sequential way of developing.

One of the sequential methods is the waterfall model. The main concept of the waterfall model is that you can not start a new activity before the previous one is complete. Projects are divided up into phases: requirements, analysis, design, development, testing, and maintenance. During the requirements phase, specific goals are set up for each of the phases. When a phase is complete and the process has entered a new phase, there is no opportunity to return to the previous phases, which can be seen in Figure 1. This implies that the waterfall model follows a linear approach where the goals must be clear and unchanged during the development [2].

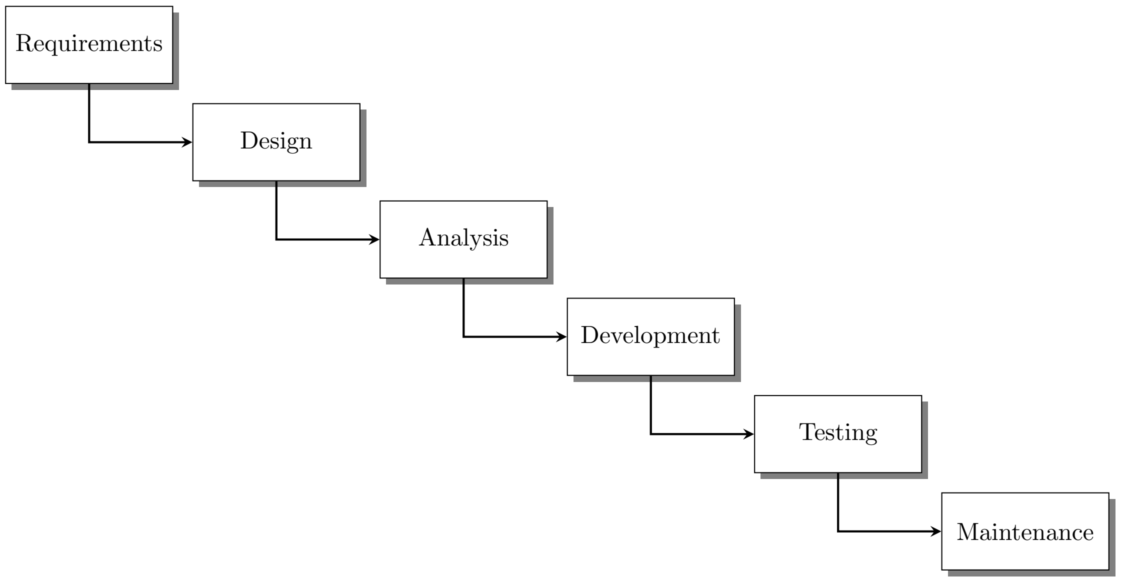

Figure 1: Waterfall model with six phases.

2.1.2 Agile development methods

If the software requirements are not established, and changes might occur during the development of the product, agile software development methods can be used. Today's volatile technology industry requires software development methods that are lighter and faster than the heavyweight approaches. One of the main concepts which form the base of agile methods is iterative development. The larger tasks are being partitioned into smaller tasks that can be refined and reworked during the whole life cycle of the software [3].

The Scrum methodology is one of the most popular agile methods. There are several people with different roles involved when working with Scrum. The Scrum process starts with the product owner who wants to create a product and creates a prioritized list with the features to be included. The Scrum master is responsible for the development team and guides them. During the start of the sprints[1], a sprint planning meeting is held by the Scrum master. The team decides on which backlog items they want to complete this sprint. Daily meetings are also held to make sure that the project progresses healthily. At the end of a sprint, the stakeholders have an opportunity to review the product and give feedback [4].

2.1.3 Comparison of traditional and agile software development methods

As mentioned in Section 2.1.2, agile methodologies focus on iterative development, whereas traditional methodologies focus more on planning and documentation. With the use of traditional methodologies proceeding to the next phase can be problematic, because every requirement needs to be fulfilled. Agile methodologies are designed to make projects more adaptable to change and documentation is not as important. This makes agile methodologies suitable for dynamic and evolving projects, as they allow for flexibility and responsiveness to changing requirements.

2.2 Software development practices

The project is organized and developed with the use of certain methodologies. Smaller tasks in the project, such as implementing certain functions, can also be organized and solved with different software development practices. Some of the practices are:

- Refactoring.
- Pair programming.
- Iterative development.
- Test-driven development (TDD).
- Daily scrum.
- Retrospectives.
- Communication.
- Planning.
- Estimation.

The purpose of this report is to evaluate iterative development.

2.3 Iterative development

There are several different approaches to building software, and one of them is iterative development. With iterative development, the project is divided into smaller sub-projects which are called iterations. The iterations are composed of different activities which help the team reach their goal. These activities are similar to the activities in the waterfall-model, which can be seen in Figure 1. The goal of an iteration is a demo that is tested and integrated with the system that will be released later [5]. Each iteration is therefore supposed to contribute with a functional increment to the final product.

Figure 2 displays iterative development with the same phases as the waterfall-model. It is important to note that the phase *Maintenance* is not always performed in each iteration. The figure includes the important concept of *Increment* which is displayed by the dotted arrow. Each iteration thus builds upon the previous ones.

[1]Sprint: Period where the development team collaborates to create a functional increment on the product.

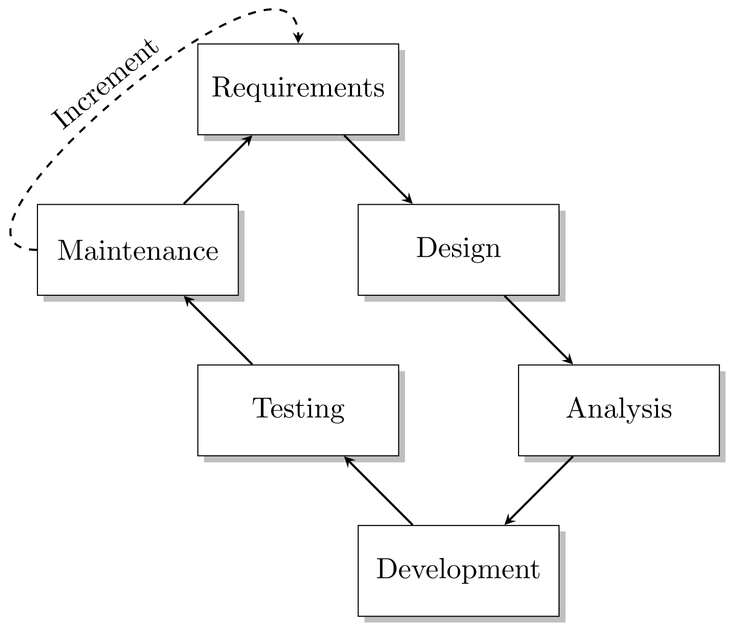

Figure 2: Iterative development with six phases.

2.4 Project presentation

The objective of our project was to create a 3D horror game where two players can play together. They have been exploring the city, and they fell into the catacombs. Their mission is to get out of the catacombs alive. To get out they need to solve multiple puzzles and avoid the ghost which wanders around. Inside the catacombs, the players are equipped with a flashlight and a camera. They can use the camera to scare the ghost, and the flashlight helps them be sane. In certain locations, pills and batteries can be found which the players can utilize to their advantage.

The game has been developed with the use of Unity which is a cross-platform game engine. Games created with Unity are written with the programming language C#. The main functionality of the game is to get friends together and give them an enjoyable time. To enhance realism, we developed the game in a 3D format. Therefore, it was necessary for us to create models and assets to achieve the visual requirements. The models and assets were created with Blender which is a graphics tool. The functionality of the players and puzzles was written in C#. This project was conducted with the use of Scrum which is explained in Section 2.1.2.

3 Research Method

3.1 Research design

The research will be conducted along with our project, and it will be divided into three phases. Figure 3 showcases how our research will be conducted.

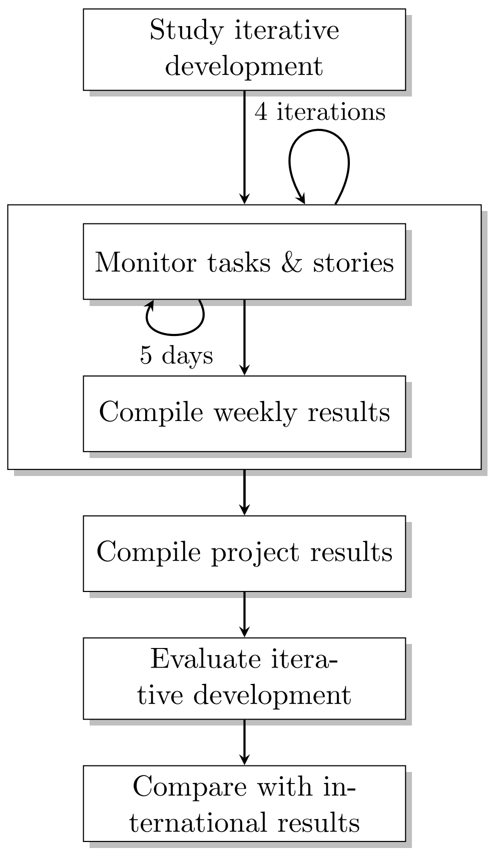
Study iterative development
4 iterations
Monitor tasks & stories
5 days
Compile weekly results
Compile project results
Evaluate iterative development
Compare with international results

Figure 3: Research method phases.

3.1.1 Phase I: Studying iterative development

Studying our chosen practice, iterative development is the first phase that is included within our research method, which can be seen in Figure 3. This phase is about getting comfortable with the concept of iterative development and understanding it so well that we could proceed to plan ahead for future phases. We got over this phase by studying literature on the subject. We agreed upon reading research papers for us to fully understand the real definition of the practice because the ordinary sources that come up after have searched for our chosen practice can be somewhat misleading.

3.1.2 Phase II: Conducting our project and gathering data

Conducting our project and gathering data is the second phase of our research method and can be seen in Figure 3 shows this phase holds onto two boxes where one box monitors tasks and stories, while the second box compiles weekly results. This was basically done by writing notes about the group's current status of tasks and stories. The note-taking was done every day before everyone left the working space, so we could conclude what had been done for each and every day, meaning that this phase was iterated five times a week, for four weeks. All the data were collected in a Google Docs document, where a well-organized table with associated headings was found. The data collected, under four different headings, for each and every day, were the number of tasks that one has started with, the number of tasks that have been finished, the number of tasks that have been added under that specific day, and lastly the number of tasks that have been removed. More data was also collected by taking raw notes from the internal sprint review and retrospective meeting, at the end of each sprint, to conclude if the team actually has provided real value and bringing progress to the project as a whole. With this data, we can later conclude the effectiveness of ending started tasks. At the end of every week, we compiled the data collected throughout the week in a weekly table, for us to later differentiate the different sprints.

3.1.3 Phase III: Analyzing and comparing the collected data

The last phase of our research method is to analyze and compare the collected data. This was possible because we now had collected all raw data for the different days and the four different sprints. It was then time to compile all the data into a final result. With the final result, we now had to evaluate the iterative development of ours. In order to evaluate something, one should have some evaluation criteria to compare the results to.

3.2 Evaluation criteria for iterative development

The following criteria are a central part of iterative development. We have taken them into account when conducting our project.

3.2.1 Time and resource management

This criterion refers to the time and resource management of all the tasks that were brought up during the sprint planning meeting, which occurs at the beginning of every sprint. In this report, we will evaluate whether the team's ability to estimate and plan tasks according to the resources available is sufficient enough for achieving the planned objectives.

3.2.2 Adaptability

Agile software development methods have a heavy focus on reacting to changes as mentioned in Section 2.1.3. This criterion will take the adaptability of the team and project into account. The product owner might have urgent stories which need to be implemented right away, this is a part of adaptability. The team should also be able to handle bugs that are discovered at a later stage in the project.

3.2.3 Project progress

This criterion refers to the progress made in each iteration, from a quality perspective, whereas the criterion, *Time and resource management*, refers more to the quantity perspective of the tasks with numbers, where this criterion assesses whether or not the project is moving forward and each iteration is bringing value. This can be assessed by the whole group's reflection at the end of the week together with the product owners last words, if the development is under progression towards the end goal.

3.3 Limitations

Conducting research on iterative development during a four-week project has limitations. The timeframe for the research is short. This means that the sample sizes will be small and might limit our conclusions because of the lack of accurate data. The short timeframe will also give us a lack of longitudinal perspective. External factors such as team dynamics, complexity, and stakeholder influences will not be taken into account. Results could thus differ if we worked with another team or did another project.

4 Project Results

All data collected from the project was done by taking notes and filling in tables with the right amount of tasks within three different headers. The data collected must provide answers to the evaluation criterion that we are going to examine. For the first criterion, the data under the headers of *Started* and *Done* provides us with information about how many tasks could be initiated along with the termination of those. For the second criterion, *Adaptibility*, there are two rows where one can see if the team had to adapt and add or remove tasks that were unplanned from the sprint planning meeting. The last criterion, *Project Progress*, refers to if the current iteration has brought value to the project as a whole. The data for that was collected by taking raw notes from the internal sprint review and retrospective meeting, at the end of each sprint. The product owner was the person with the final word to say if we gained value or not.

4.1 General results

Table 1 below illustrates the results from week one. During the first day, the team only got started with one single task, which was to learn Unity, watch tutorials and etc. Nothing got done, taken away, or added that day. On the second day, the team started to assign tasks to themselves and work with the fundamentals that were needed for the project to succeed, six additional tasks got initiated that day and three got finished. On the third day, we initiated three more tasks, got done with five tasks, and added two tasks to our planning for the week. On the fourth day, we initiated seven new tasks, got done with five tasks, and added another task. On the last day of the week, the team had other work responsibilities that did not require them to work primarily in Unity and code the actual game, thus no tasks were completed. We initiated 17 tasks and managed to finish 13.

Tasks	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Started	1	7	10	17	17
Done	0	3	8	13	13
Added	0	0	2	1	0
Removed	0	0	0	0	0

Table 1: Management of tasks during the first iteration.

Table 2 illustrates the results from week two. This week started off by initiating eight tasks and finishing six tasks. We also finished some tasks from the previous iteration which were not done in time. On the second day, an additional 13 tasks got initiated, six new tasks were finished, three new tasks were added and four tasks got removed. On the third day, we initiated three new tasks to the prior day, six tasks got finished and two were added. This week focused on using the map layout and the movement mechanism from the past week to add additional game mechanics like a co-op feature, a limited flashlight, a Polaroid camera, and some other things.

Tasks	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Started	8	21	24	28	28
Done	6	12	18	22	22
Added	0	3	2	0	0
Removed	0	4	0	1	0

Table 2: Management of tasks during the second iteration.

Table 3 illustrates the results from week three. There is one less day in this table and that is because one of the days was a holiday. We started off the week with seven tasks, four got finished and one got removed. On the second day, we initiated an additional four tasks and got done with two new tasks, and a task was also removed during this day. On the third day, we did not initiate new tasks instead, we worked with the ones we had prior and finished them off so we could end the week according to the plan. We finished the tasks that we started, meaning that we finished off with five tasks that day. The fourth day is always mostly set for other things to do and therefore no progress in tasks got done that day, but we still managed to finish off according to plan.

Tasks	**Day 1**	**Day 2**	**Day 3**	**Day 4**
Started	7	11	11	11
Done	4	6	11	11
Added	0	0	0	0
Removed	1	1	0	0

Table 3: Management of tasks during the third iteration.

The fourth and last iteration was very crucial for us, we needed to integrate everything we developed into our map. During the first day of the iteration, we started with eleven tasks and we did not finish any of them. On the second day, we needed to remove three tasks so that we would be able to complete the project. On the second day, we started with five new tasks and finished ten tasks. No new tasks were added during the second day. 14 new tasks were started during the third day of the iteration, we also completed an additional of nine tasks. One new task was added on the third day. On the fourth day, we started with eight new tasks and managed to complete seven of them. On the last day, we completed the task which was left over from the fourth day of the sprint. This data can be seen in Table 4, down below.

Tasks	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Started	11	16	30	38	38
Done	0	10	19	37	38
Added	0	0	1	0	0
Removed	0	3	0	0	0

Table 4: Management of tasks during the fourth iteration.

Table 5 illustrates the results compiled from all the different weeks. On the first sprint week, iteration one, the team initiated 17 tasks, got done with 13 of them, and added three tasks. On the second sprint week, iteration two, the team initiated 28 tasks, got done with 22 of them, added five new tasks, and removed five tasks. On the third iteration, with one less day to work with because of a holiday, we only initiated eleven tasks and finished all of them, two tasks got removed during this iteration. On the last iteration, the fourth sprint week, we had initiated 38 tasks, and why this number is so much higher than in prior weeks because we chose to divide stories into smaller and smaller tasks. We managed to finish all of the tasks. One new task was added during the iteration, and three tasks were removed.

Tasks	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**
Started	17	28	11	38
Done	13	22	11	38
Added	3	5	0	1
Removed	0	5	2	3

Table 5: Management of tasks during the four iterations.

The project spanned over four weeks, and each week represents one iteration. The team managed to start 94 tasks. During the iterations the team came up with new ideas and the product owner had new requirements. This resulted in us removing ten tasks and adding nine new tasks to work on. We managed to complete 84 tasks, this data can be seen in Table 6.

Tasks	**Total**
Started	94
Done	84
Added	9
Removed	10

Table 6: Management of tasks during the whole project.

The tasks which we worked on during the iterations were based on the stories which the product owner set up for the team. During the first iteration, we decided to work with four stories. We managed to complete all four stories. The second iteration had a few more bugs than the first iteration. We opted to finish seven stories but we only managed to complete five stories successfully. The third iteration was one day shorter, therefore we decided to only work with three stories. The team managed to complete every story in time. During the last iteration, we decided to only work with four stories. We needed to focus on integration and bug fixes. We managed to complete all four planned stories. The data is displayed in Table 7.

Stories	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**
Planned	4	7	3	4
Done	4	5	3	4
Added	0	0	0	0
Removed	0	0	0	0

Table 7: Management of stories during the four iterations.

Table 9 illustrates the story points results from all the iterations. In the first iteration we had planned to finish stories worth 128 story points, but we could manage to work out 150 story points instead. In the second iteration, we had planned for 158 story points but managed to work out 206 story points. During the third iteration, we had planned for 150 story points to be done by the end of the week and we worked according to plan. During the last iteration, we had planned for 100 story points but managed to instead work out 128 story points. During all of these weeks, we had either worked according to plan or had underestimated our abilities, therefore we always could do more than was planned.

Hours	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**
Planned	128	158	150	100
Completed	150	206	150	128

Table 8: Time estimation during the four iterations with story points.

In total the team planned to complete 536 story points. The team managed to complete 634 story points which is an additional 98 story points completed. This can be seen in Table 9.

Hours	**Total**
Planned	536
Completed	634
Extra	+98

Table 9: Time estimation during the whole project with story points.

We needed an adequate way to measure how successful the iteration was and if an increment has been made according to the last evaluation criteria which is mentioned in Section 3.2.3. The way we decided to measure satisfaction and increment was to collect opinions from the product owner and the team during the internal reviews. We get two different views on if an increment has been made. We created a "scale of satisfaction" which represents if an increment has been made or not. The scale ranges from 0% up to 100%, where 0% represents no satisfaction and thus no increment. 100% on the scale represents total satisfaction and a full increment has been made.

During the first iteration, the product owner had the satisfaction of 93%, while the team's satisfaction was 96%. The second iteration resulted in the product owner being satisfied up to 85% and the team was satisfied up to 91%. Both the team and the product owner had the satisfaction of 92% during the third iteration. The fourth and last iteration resulted in the product owner being satisfied up to 97% and the team up to 95%. This data is displayed in Table 10.

Satisfaction	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**
Product owner	93%	85%	92%	97%
Development team	96%	91%	92%	95%

Table 10: Satisfaction during the four iterations.

4.2 Words from product owner

During the internal sprint review our product owner shared his own thoughts on the iteration. We decided not to share the thoughts of the team because it consisted of nine people. This would result in ignoring some thoughts and that would not give a fair picture of the iteration.

4.2.1 Iteration 1

At the first internal sprint review, our product owner said, *"This first sprint was a success for the team because it focused on learning Unity and all of its features, which the team did. The team also met the sprint target by developing and implementing first-person movement and creating a map layout for the game. In addition, the team was able to make extra elements such as a rudimentary player model, headbobbing, ladders, and doors. Even though some of the features had minor problems, I was thrilled with the outcome of this sprint because its goal and more were met."* [6].

4.2.2 Iteration 2

During the second internal sprint review, our product owner said, *"This second sprint was dedicated to implementing multiplayer and designing all player mechanics, such as a stamina and sanity system. Other player mechanics include an inventory system, an interaction system, and a camera system that allows the user to shoot Polaroid photos in-game. These systems were implemented to the bare minimum to meet the sprint goal. This is due to the multiplayer's implementation of only movement and no additional features. Furthermore, the team was able* *to make an enemy AI that only worked in single-player. With that said, I was pleased with the results of this sprint, despite having higher expectations for the multiplayer function."* [7].

4.2.3 Iteration 3

Our product owner said, *"This sprint aimed to develop multiplayer with all game mechanics. Furthermore, it focused on having a functional pause and start menu, having an enemy AI that works for multiplayer, developing all puzzles, and incorporating some of them into the game. This goal was not fulfilled since not all parts were completed. All menus and multiplayer functions were completed and met their objectives. In addition, all the puzzle mechanisms were developed and worked, though no puzzle was incorporated into the game. As a result, the sprint goal was not accomplished, nonetheless, I am still delighted with the outcome because most of the parts were met."*, during our third internal sprint review [8].

4.2.4 Iteration 4

During the fourth and last internal sprint review, our product owner said, *"This sprint goal was to bugfix all game mechanics, add sound, animations, textures, and models, improve performance, and publish the game. Because the previous sprint goal was not met, this sprint goal also includes incorporating all puzzles into the game. These objectives were completed, and the game was released in version 1.0. With this launch, I was overjoyed with the conclusion of this sprint and the finished game."* [9].

5 Analysis

The project, which we conducted with Scrum, was the first time for every member of our group to use an agile development method. This means that we are inexperienced both with time and resource management and adaptability. Project progress is also hard to measure with adequate methods. Therefore we decided to utilize Table 10, where the product owner and the team could give their own opinion on how much the project has progressed during the specific iteration. We also decided to use quotes from the product owner which would give us another perspective on how well the project has progressed.

Using all raw data collected from the project, which is shown in Section 4, one can analyze the data and reflect on that data. The first criterion to analyze is *Time and resource management*. The whole team was of course inexperienced with this sort of way of working, using Scrum, and therefore it is understandable if the team were worse at planning and estimating at the beginning of the project versus at the end of the project. Before wanting to plan what stories and tasks that should be needed, the group had to fully understand the product and the end goal, which was not quite the case for us. Not only that but even when we brought up some stories and tasks during the sprint planning meeting, the estimation of those was only based on our own personal knowledge. One can see that the results from Table 1 and Table 2 follow a pattern, where both tables result in data reporting that the team initiated more tasks than they could finish, which is a direct consequence of the inexperience that the team had. In the last two weeks, one can see in the data from Table 3 and Table 4, that they follow a different pattern compared to the first two weeks. The team got more comfortable with estimating and planning, and that resulted in more accuracy, which can also be seen as we were able to finish every single task that we had initiated.

Effort and time estimation in software engineering is something that is important. When producing a product it is natural to estimate the cost which is required to create the product. There are not many external factors that may affect the cost. But when estimating the time required in software engineering it is more complex. There are many external factors such as human, technical, and perhaps political, and their impact on the project can never be predicted. Even though the external factor exists there are several time and effort estimation models such as the *empiric non-parametric estimation model* (ENPE-model), the *expert estimates model*, and *analogue models*. The ENPE-model estimates time and effort based on data from projects which were conducted earlier. It does not utilize any mathematical formulas, instead, it uses other models such as *optimized set reduction technique*. The expert estimates model is based on the consultation of multiple experts in the field of software engineering. Analogue methods utilize analogies between the current project and previous ones [10].

When realizing our project we did not utilize any of the estimation models mentioned above. Our estimation consisted of the group members consulting each other about their previous experiences with similar problems. This is similar to the expert estimates model. The only difference is that none of us was an expert in software engineering when conducting the project. This resulted in inaccurate estimations, in the beginning at least. If we instead used the models provided we would perhaps have more accurate estimates and the iterations would have been smoother.

The second criterion to analyze is *Adaptibility*. The team worked with Scrum, an agile software development method designed to make projects more adaptable to change, and the documentation is left behind, as presented in Section 2.1.3. If we take a look at our data collected, referring to Table 1, 2, 3 and 4 one can see that we had to adapt the tasks every week, adding and/or removing tasks, that were not really relevant for the fundamental part of our product. In comparison to the time and resource management criterion, we got better for every week that passed, but for this criterion, our teams adaptability was mostly based on the product owners needs for the product, that continuously changed during the project. This was something that the developers at least could not control for most of the part, as they worked under the directions of the product owner. The reason for this poor management of the teams adaptability could be based on that we did not have stable documentation of the product, which is what Scrum advocates.

Adaptive models focus more on documentation and constructing more comprehensive user requirements. The main objective of the adaptive models is to achieve agility and adaptability. Qureshi, Rizwan Jameel, and Hussain explain this in their paper, *"One of the main objective of adaptive process model is to eliminate prominent limitations of agile process models and especially of XP."* [11]. Adaptive models thus lead to increased transparency between the group members.

If our team instead tried to adapt some of the principles of adaptive models we would have been better at adapting to change. More transparency between the group members would lead to everyone knowing what is about to happen. The changes would not be rapid and the team would have had more time to think them through. This would result in that every iteration would be more transparent and the members of the team would know if they will need to adapt to changes.

The last criterion, *Project progress*, refers to the increments in quality in the product, meaning if we made progress with the actual product, the game. To evaluate this criterion, we chose to collect data in terms of taking raw notes from our product owner at the end of every week, during the internal sprint reviews. We could not collect data in numbers for this, instead, we are going to evaluate the progress based on Section 4.2. This might not be the best way to evaluate this criterion, but we thought that opinions from the person that got the whole vision for the product in their mind would be best suitable for answering this criterion.

The first sprint resulted in success where the team actually could reach the sprint goal of implementing the fundamentals of the game, such as first-person movement, adding a character, and a map layout. The team also could implement some extra features such as ladders and doors. Our product owner said, *"I was thrilled with the outcome of this sprint because its goal and more were met."* [6]. The satisfaction of the team and the product owner was high during the first iteration. For the second iteration, the satisfaction rate dropped both for the team and the product owner, due to problems with implementing multiplayer. The team needed to rewrite scripts that were already implemented and working for single-player, this resulted in a higher workload than expected. At the third internal sprint review our product owner said *"In addition, all the puzzle mechanisms were developed and worked, though no puzzle was incorporated into the game. As a result, the sprint goal was not accomplished, nonetheless, I am still delighted with the outcome because most of the parts were met."* [8]. The sprint goal was not met but the satisfaction rate was relatively high, this is because we only had minor fixes left. The fourth and last iteration had the highest satisfaction rate of all iterations. All objectives from the previous iteration were met and the first version of the game was released.

According to Schwaber, Scrum teams are required to make a functional increment each iteration. The increment must be shippable because the product owner decides if the functionality needs to be implemented directly. Schwaber's paper does not give any metrics on how to measure an increment. Schwaber said *"This requires that the increment consists of thoroughly tested, well-structured, and well-written code that has been built into an executable and that the user operation of the functionality is documented, either in Help files or in user documentation. This is the definition of a "done" increment."* [12].

To measure something which is "thoroughly tested", "well-structured" and "well-written" is something that every team should have definitions for. When starting our project we did set up a "definition of done" which took some of these factors into account. The problem is, something can be done but what value/increment does it bring to the product? In the end, it is the product owner who decides if the value has been brought to the project. According to our data, which can be seen in Table 10 and the quotes from our product owner, he was satisfied during the whole project. According to him, increments have been made during iteration.

6 Conclusion

In conclusion, our practice, *Iterative development*, is effective, but there is still a lot to improve with our project. The team was inexperienced with this type of project and it was the first time for everyone using Scrum. The team did manage to build a finished product that the product owner was happy with, not only that but the team learned to manage their own time and resources throughout the project, which is nice to see. The team was also able to adapt themselves, but not quite well, we always had to change something, and that may have been caused by not using some adaptive models. The collecting of all data worked pretty fine, but we were still quite limited on that front, mostly for evaluating *Project progress*. Until next time it would have been better if we would had documented the vision for our product in the beginning, so it would have been easier to assess and evaluate the progress on the product with some concrete comparisons, of what we were supposed to do versus what we actually did. We can still conclude that this project and report was a success, we were able to learn a lot and we can assure that we are far more experienced today than four weeks ago.

7 References

[1] S. Al-Saqqa, S. Sawalha, and H. AbdelNabi, “Agile software development: Methodologies and trends.”, *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, pp. 247–248, 2020.

[2] A. A. Adenowo and B. A. Adenowo, “Software engineering methodologies: A review of the waterfall model and object-oriented approach”, *International Journal of Scientific & Engineering Research*, vol. 4, no. 7, pp. 427–434, 2013.

[3] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile software development methods: Review and analysis*, 2017. arXiv: `1709.08439 [cs.SE]`.

[4] S. Sharma and N. Hasteer, “A comprehensive study on state of scrum development”, in *2016 International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, 2016, pp. 867–872. DOI: `10.1109/CCAA.2016.7813837`.

[5] C. Larman, *Agile and iterative development: a manager’s guide.* Addison-Wesley Professional, 2004.

[6] V. Schütt, *Communication during internal sprint review*, Conversation with the author on April 21, 2023, Apr. 2023.

[7] V. Schütt, *Communication during internal sprint review*, Conversation with the author on April 28, 2023, Apr. 2023.

[8] V. Schütt, *Communication during internal sprint review*, Conversation with the author on May 5, 2023, May 2023.

[9] V. Schütt, *Communication during internal sprint review*, Conversation with the author on May 12, 2023, May 2023.

[10] J. Živadinović, Z. Medić, D. Maksimović, A. Damnjanović, and S. Vujčić, “Methods of effort estimation in software engineering”, in *Proc. Int. Symposium Engineering Management and Competitiveness (EMC)*, 2011, pp. 417–422.

[11] M. R. J. Qureshi and S. Hussain, “An adaptive software development process model”, *Advances in Engineering Software*, vol. 39, no. 8, pp. 654–658, 2008.

[12] K. Schwaber, *Agile project management with Scrum.* Microsoft press, 2004.