CROSS-DISCIPLINARY COMMUNICATION IN GAME DEVELOPMENT:
How does modifications to agile methodologies affect communication across disciplines?

Master Degree Project in Informatics
with a Specialization in Game Development

Two year Level 30 ECTS
Spring semester 2023

Selin Kaya

Supervisor: Per Backlund
Examiner: Mikael Johannesson
Abstract

Many software methodologies are being used for the development of video games. However, there are very few studies that cover how these methodologies are deployed and what they affect in the game development process. This study aims to explore the connection between agile development modifications applied in video game studios and their effect on cross-disciplinary communication. The research was conducted in three parts; talks of professional game developers were analysed, a field study in a studio was conducted, and a survey was designed to gain more insights from game developers around the world. After obtaining the results in three parts, a content analysis was carried out to find patterns in agile development that affect communication across disciplines. The found patterns were the overlapping work of developers, the amount of cross-disciplinary meetings, and doing experiments to find a methodology that best fits the development team. This research contributes to the field of game development with a focus on the impact on communication between different disciplines with the usage of agile development.

**Keywords:** video game development, disciplines in game development, agile methodology, scrum framework
# Table of Contents

1 Introduction

2 Background

2.1 Video Game Development
2.1.1 Disciplines in the Game Industry
2.1.2 Indie, AA, and AAA
2.2 Project Management Framework
2.3 Game Development and Software Engineering
2.4 Agile Development
2.5 Scrum
2.5.1 Scrum in Game Development
2.5.2 Positive Impacts of Scrum Methodology
2.6 Issues in Game Development

3 Problem

3.1 Method
3.1.1 Content Analysis
3.1.2 Field Study: Participatory and Direct Observation
3.1.3 Survey

4 Results

4.1 Game Developer Talks
4.1.1 Improving the Culture of Critique: Communicating Across Disciplines by Jeff Hesser
4.1.2 Managing Conflict on Small Teams by Rebekah Saltsman
4.1.3 EPIC Sync: Wrangling the Work of Highly Interdependent Dev Teams at Bungie by Linda Fane
4.1.4 Taming the Chaos: Lessons in Project Management by Clinton Keith
4.2 Field Study Results
4.2.1 Participatory Observation
4.2.2 Direct Observation
4.3 Survey Results
4.3.1 Pilot Study Feedback
4.3.2 Basic Information Results
4.3.3 Development Methodologies Results
4.3.4 Communication Across Disciplines

5 Analysis

5.1 Game Developer Talks Analysis
5.2 Field Study Analysis
5.3 Survey Analysis
5.4 Overview of Patterns Found

6 Conclusions

6.1 Summary
6.2 Discussion
6.2.1 Validity of Previous Research
6.2.2 Ethical and Social Aspects
6.2.3 Limitations
6.3 Future Work

References
1 Introduction

The games industry is a rapidly growing industry. In the last five years, the number of game companies in Sweden has shown an increase of \( \%80 \), from 513 studios to 939 studios in 2022, while the world market of games is estimated to reach a value of 212.4 billion dollars by the year 2026 (Dataspelsbranschen, 2023). Despite the popularity of games, apart from game developer postmortems and conference talks, getting insights about the practical side of game development is very limited, since it is rarely studied by researchers (Engström, 2020).

Many disciplines come together and form development teams in the games industry (Rogers, 2014). Compared to non-game development, game development is messy (Whitson, 2020), chaotic (Keith, 2020), and less strict when it comes to requirements (Murphy-Hill, Zimmermann and Nagappan, 2014), because as a game develops many ideas come in and go (Saltsman, 2017), and in its core, game developers tend to focus on delivering a fun experience for their audience (Hodent, 2021), which are commonly referred as players. Since game development tends to be more creative compared to other software development, more diverse teams are generally formed. However, similar to other developers, game developers also utilise development strategies to plan and establish a workflow.

With the increase in the usage of iterative development strategies in the software industry (Kristiadi et al., 2019), such as agile, video game development studios are shifting from separating the disciplines in silos to forming smaller cross-disciplinary teams that focus on separate features of the game (Fane, 2018). The flexibility and scalability of iterative practices, allow development teams to experiment with different practices to find the best methodology that fits with them.

Nevertheless, even with the correct application of development strategies, developers can be faced with issues, regarding planning, technical debts, and crunch (Petrillo, et al., 2008; Keith, 2010; Keith 2020). A typical solution for the mentioned problems from developer postmortems is to improve in planning and estimation, however, Whitson (2020) argues that the occurrence of the issues is related to social norms, rather than technical.

With the motivation to explore a connection between development strategies and social aspects in game development, the goal of this study is to find patterns in modifications made to agile development strategies that pose an impact on cross-disciplinary communication in video game studios.

This study is structured as follows: chapter 2 covers an introduction to the video game industry and the iterative software methodologies generally used, chapter 3 states the research question and the methods used throughout the research, chapter 4 presents the results found, chapter 5 showcases the content analysis done on the results to find patterns, and chapter 6 concludes the paper.
2 Background

This research is closely related to game development and software management frameworks. This chapter starts with an introduction to video games and the games industry, followed by an explanation of project management frameworks and comparisons between game development and software engineering. Finally, the last two sections of the chapter, explain agile development and one of the most commonly applied methodologies, Scrum.

2.1 Video Game Development

The first iterations of video games became present in the 1950s. They were created in computer laboratories of large universities (Rogers, 2014), solely by programmers. Games back then were played on huge computers by a handful of people (Rogers, 2014), and they were not commercially accessible until the 1970s.

During the late 1970s, personal computers (PCs) started to become popular as a device used for both video game development and playing video games (Rogers, 2014). These early games made on computers were stored on cassettes, which later turned into floppy disks, and once the computer hardware became more advanced, CDs and DVDs (Rogers, 2014).

In the early 2000s, with the Internet becoming more accessible to people around the world, digital marketplaces for games were being created which allowed studios to digitally distribute their games. Steam (2023) a client that was first used to update games released by Valve (2023), is now one of the most popular services for purchasing video games with a monthly active user count of 132 million users (Balasubramanian, 2022).

In 2023, it is said that more than three billion people have had experience playing digital games (Isfe, 2023). Having reached a high percentage of the global market, video games are accessible on various platforms, from computers to phones.

2.1.1 Disciplines in the Game Industry

Game development is closely related to software engineering since the result of the processes is executable software. Different from software engineering, there are many different disciplines other than engineering that participate in game development (Rogers, 2014). Additionally, games tend to focus on delivering players a fun experience and setting that as a goal for the developers (Hodent, 2021).

Engström (2020) connects the Krebs Cycle of Creativity (Oxman, 2016), shown in Figure 1, with video game development since all the modalities shown in the cycle are related to the different disciplines that make up a video game development team.
People who create video games are known as developers or development teams (Rogers, 2014). Games can be created by a group of developers or even solo.

The disciplines can be mainly categorized as (Bay, 2023):

1. Design
2. Art
3. Programming
4. Audio
5. Writing
6. Testing
7. Production
8. Management

**Figure 1** The Krebs Cycle of Creativity (Oxman, 2016).
The given categories are very broad and can get more detailed when mentioned about the professions they are related to. To be able to understand how the disciplines communicate with each other, more detailed descriptions of the mentioned categories and the disciplines will be mentioned. These descriptions are generalised and may not apply to all the studios. Every studio has a different way of working, and recruiting and the disciplines may differ in technicality and responsibilities.

Designers in video game studios come up with ideas and rules to make a game. Game designers usually focus on creating the mechanics, systems, and rules of the game. On the other hand, level designers are responsible for creating levels and environments for the player to interact with throughout the game. User interface (UI) designers, create the interface that shows the information about the game that the player needs to keep track of and know about more efficiently. AAA studios might have more in-depth designers such as combat designers, who specifically focus on the player’s combat experience with an AI or human opponent, and how the player’s flow will go.

Artists are responsible for creating the game’s art, whether it is 2-D, 3-D, or animations. Concept artists do sketches of what the game characters, worlds, and enemies will look like in the game (Rogers, 2014). Visual effects artist creates particle effects to add details to the environment, such as dust particles, and water waves. 3-D modellers create character and environment models. Animators animate characters by binding the joints (points where the bones of a character meet) with the 3-D model so that the character’s model moves in a certain way. Texture artists paint surfaces onto 3-D models that give the models colour and texture. Programmers write codes that give functionality to the assets that have been put in this game (Rogers, 2014). Programmers need to have a good understanding of mathematics, physics and the systems that make up the game to add functionality to it. For example, gameplay programmers focus on making the gameplay elements come to life. This can be a building mechanic in a simulation game, or player movements in an action game, and so on. Tool programmers can work on improving the game engine’s usability by creating new add-ons inside the engine to help other team members work more efficiently (Rogers, 2014). Graphics programmers work mainly on making the visuals in the game work efficiently while producing good-quality results. There are even programmers that work with the audio department to trigger the music and the sounds at the right cues.

People working with audio can be separated into two different disciplines; composers and sound designers. Composers write the music score of the game. The music can be created in many ways; some examples given may be, using a keyboard or orchestrating a live orchestra. On the other hand, sound designers create sound effects in the game. From creature growls to door creaks, to user interface clicks, sound designers are responsible for creating the audio cues for different elements of the game.

Writers work on the story of the game. They write the dialogue between characters and provide good and clear instructions for the tutorials. Rogers (2014) mentions that game writers are involved in the game’s development process later than the other disciplines, however, that understanding has started to change in the game industry recently. Having a writer earlier in the process has proven to show that they aid in directing the flow of the game’s content. From what has been encountered in my work, they know the characters and the events in the game by heart and are mainly responsible for creating quests that the player will experience during the gameplay.

The testers’ main goal is to keep the game bug-free and playable. They are responsible for testing out the same levels over and over again to find bugs that can cause crashes and break systems that can make the game unable to progress. Once the bugs have been discovered, they have to have good communication and documentation skills to report these problems to the development team (Rogers, 2014). Quality assurance (QA) is the discipline that checks over
the builds of the game and gives the green light whenever a game has come to a quality where it is ready to be submitted or published.

Producers oversee the entire game development team (Rogers, 2014). They have responsibilities in hiring and building teams, writing contracts for any deals with publishers or investors and becoming the team’s representative to upper management and outside resources. These responsibilities are examples, and they can differ across different-sized studios.

Since producers have a lot to manage, there are usually studio coordinators who help out with day-to-day tasks. As said before, a producer’s work can differ amongst many studios, some producers are not a part of the development team and they are within the publishing team that helps out the development studio when in need. Rogers (2014, p. 21) says “Regardless of how helpful producers can be, some development studios consider producers to be an unnecessary part of the development. Others feel that producers should not have any creative control, just manage the game’s production and schedule.”. The state of the producers varies within the industry (Rogers, 2014).

Aside from the development team, other departments can get involved with a studio, that can be publishers. Next to publishers, similar to producers, some managers have roles that act as an intermediary between the development studio and the publishing company. Publishers provide the funding for game development teams, manage the production, handle legal issues, and provide public relations and marketing for the game. Product managers work similarly to producers. They work with the development team and manage them based on the agreed production schedule (Rogers, 2014). They help in determining the production’s priorities, reviewing and approving milestones, and making payments to the studio. Product managers also help in getting in touch with the licensors and update them on the development. Creative managers help teams create and develop games in terms of design and narrative, write game pitches and work with licensors to create game concepts (Rogers, 2014).

There are still many careers in video game development that have not been covered here, and the description of these disciplines varies across the game industry. They are dependent on the studio’s policies, size, location and many more criteria. However, this summary of the disciplines will help in understanding what might cause the miscommunication between them and ways to improve it.

2.1.2 Indie, AA, and AAA

The industry has become familiar with some ways to describe video game studios depending on their development budget, marketing sources, team size, production values, future franchising and more criteria (Karthikeyan, 2022). Development processes not only differ across disciplines, but they vary between studios as well. This difference can be seen in the games developed by a small amount of developers or more than thousands of developers (Engström, 2020).

Independent games, more commonly known as, indie games are referred to as games created by indie companies (Karthikeyan, 2022). Indie companies are not owned by a parent organization that provides creative direction (Karthikeyan, 2022). In addition, indie companies are generally small-sized studios; from solo developers to ten to twenty people. Some examples of indie games are Stardew Valley (2016), Hollow Knight (2017) and Hades (2020).

Double-A (AA) studios are larger than independent studios in terms of employee count but have a smaller budget compared to bigger companies (Karthikeyan, 2022). Different from indie studios, they are usually backed by an investor, or a publisher (Karthikeyan, 2022). The team size of AA studios is usually under fifty employees. Some examples of AA games are Life is Strange (2015), Hellblade: Senua’s Sacrifice (2017) and Outer Worlds (2019).
Lastly, Triple-A (AAA) studios are companies that have the largest budget for both development and marketing (Karthikeyan, 2022). The development budget is usually higher than fifty million dollars and the team consists of hundreds or thousands of developers. These studios can consist of sub-studios for their projects. An example can be given for the development of Assassin's Creed (2007), where the publisher and parent company was Ubisoft but the development studio was one of the sub-studios of Ubisoft in Canada named Ubisoft Montreal. Games developed by AAA studios tend to be a huge success, that allows the studio to turn them into a series of games (Karthikeyan, 2023). Some examples of AAA games are the Assassin’s Creed series (2023), Final Fantasy games (2023), and Uncharted series (2022).

A game’s development is affected by many factors such as the budget and the size of the studio. However, without plans and guidance, a project’s development can get messy fast (Keith, 2020). To make the most of the development time and budget, studios tend to establish frameworks that help management and the workflow.

### 2.2 Project Management Framework

Jain and Suman (2018, p. 1) explain project management and the framework as:

> Project management is a discipline that governs skills, knowledge, tools, and techniques that can help in fulfilling project requirements towards successful software development. A project management framework consists of stakeholders, knowledge areas, tools, and techniques for managing, monitoring and controlling projects.

Jain and Suman (2018, p. 1)

To make sure that the development is on track, several keywords are used to oversee the progress; scope, time, cost, quality, risk, human resource, communication, procurement, stakeholder, and integration management (Jain and Suman, 2018). Having an effective project management technique for the development team increases productivity, the end quality of the product, and reliability towards the team, while reducing the development time and cost (Jain and Suman, 2018).

Projects that are on a global scale can go through many challenges in terms of project management. Jain and Suman (2018), go through some of the project management challenges in global software development in terms of communication, coordination, collaboration and control problems.

However, game development is fairly different from just software development. As mentioned in the previous section, there are more disciplines involved in game development than software engineers and the process of developing a game differs from developing another software.

### 2.3 Game Development and Software Engineering

Game development and software engineering are different from each other. In a study conducted by Murphy-Hill, Zimmermann and Nagappan (2014), the researchers interviewed developers with experience in both game development and non-game development. The interviewees placed a significant emphasis on the differences between the requirements of making games and than in other software (Murphy-Hill, Zimmermann and Nagappan, 2014). Same as the interviewees in this study, game studies authors, and game developers do agree on the requirement of a game, that is to be fun (Murphy-Hill, 2014; Rogers, 2014).

Games are engaged differently compared to other software. For example, engaging with an ATM takes a couple of minutes to get our task done. There are limited things a user can do with an ATM, and it does not take hours to complete the process; however, players can play
games for several hours in one sit-down. The user experience (UX) requirements for games and non-games are different (Murphy-Hill, Zimmermann and Nagappan, 2014).

Players want games to give unique experiences in gameplay, whether it be a new story or mechanic; they look for new experiences (Murphy-Hill, Zimmermann and Nagappan, 2014). In regular software development, such as computer operating systems, people generally do not want things to change, because they have already given the effort of learning it from scratch, and tend to get annoyed by the small changes that do not continue in the newer updates.

Testing done on games is rarely automated and is usually done with the addition of a QA service, with people going over the levels again and again. Games tend to have a lot of states where it is user-driven (Murphy-Hill, Zimmermann and Nagappan, 2014), so it is important to see how players take on the game and if the developers’ intentions match with what the players are doing. Hodent (2018) also refers to game developers conducting UX tests, in which the developers get to observe the player while they test out a feature, which does not usually happen in traditional software development, since the testing is mostly automated. In games, it is not easy to write tests that will act or react like a human (Murphy-Hill, Zimmermann and Nagappan, 2014).

Murphy-Hill, Zimmermann and Nagappan’s (2014) survey study on Microsoft developers, who worked in games, in Office (2023) and other software. The results of the survey showcased that game developers usually have less clear requirements than non-game developers, non-game developers tend to follow the software methodologies to its rules more than game developers, creativity is often more valued in game development and most importantly, the ability to communicate with disciplines other than engineering and having a more diverse team is valued more in game development (Murphy-Hill, Zimmermann and Nagappan, 2014).

Compared to traditional software development, games tend to be a more engaging product even before they are released. For games to do well, they go through testing cycles where they need to match their requirements with their target audience. These processes vary throughout different budgeted studios as well, some games tend to repeat a similar gameplay with a graphical update and minor changes, on the other hand, some other studios push out games that are completely different story and gameplay-wise. Although there are significant differences between traditional development and game development, some software methodologies are used commonly in game development as well.

### 2.4 Agile Development

Traditional software development methodologies, such as the waterfall model are used in game development however due to the unpredictable, chaotic nature of the development, many developers choose the iterative way of development (Koutonen and Leppänen, 2013). Koutonen and Leppänen (2013) state that by using iterative development, the developers can test out ideas before fixating on getting one feature in with a shippable quality. By testing out how the development goes, developers allow themselves to see and figure out the fun in the game.

Studios tend to use agile principles as their core for an iterative way of development. However, not to be confused, agile development is not a framework, it consists of several values and principles to help the developers adapt to changes quickly (Atlassian, 2023a).

The Agile Manifesto written by Beck, et al. (2001) contains four main values and 12 principles that formulate how the framework is structured.
The four values are:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

The 12 principles are:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

These values suggest that the process of making a product becomes a cycle of adaptive development, with the user always in mind. It reduces unnecessary time spent on documentation and focuses on the progress of the product itself. The manifesto intended to establish a common understanding of a work environment that was built on respect, support, trust, and collaboration. With this understanding, the team is put first, and it is the trust that leads the team to make work happen.

Koutonen and Leppänen (2013) surveyed Finnish game studios about how game studios use agile methods and practices. As a result, they found that the scrum model was the agile method that was highly used by the studios. Similarly, another study conducted by Kristiadi et al. (2019) found that %50 use the scrum methodology for studios that use agile development.

### 2.5 Scrum

Scrum is an agile development framework that splits up the work into short cycles called sprints. Sprints are periods of time where the developers work on the tasks at hand (Atlassian, 2023c).

During a sprint, usually in the mornings, the team does a daily scrum or a daily stand-up, where each team member goes over three questions to inform the others about how their sprint progress is going. These three questions are:

1. What did I do yesterday?
2. What am I going to work on today?
3. Do I have any blockers or questions?

The daily stand-up is called a “stand-up” because this method is typically used while the whole team is standing up to keep the meeting shorter.
A sprint review is done usually at the end of a sprint to go over what has the team completed during the sprint (Atlassian, 2023b). A sprint review is a meeting where the team gathers up to discuss which planned tasks have been “done” and get feedback on the work. The meaning of “done” for a task can vary between different studios; some companies measure the working hours as a definition of done, and the more hours spent working means more of the tasks get done, other companies say that a task is “done” by seeing if the work is in the build of the game and working as intended (Keith, 2020).

The sprint retrospective, similar to the sprint review, is typically done at the end of sprints to evaluate what went well, and what can be improved for the upcoming sprint (Atlassian, 2023b).

The sprint planning is a meeting where the list of features for the development team to implement for an upcoming sprint gets decided on.

Additionally, there are three key components in Scrum to keep track of the tasks of the project; the product backlog, sprint backlog and sprint goal.

The product backlog is the list of work that needs to be done (Atlassian, 2023d).

The sprint backlog is the list of features that the development team will work on in a sprint. Some changes can occur with the backlog during a sprint, but the core goal of the feature stays the same.

The sprint goal is the end product of a sprint. Aligned with the sprint review, the sprint goal showcases what has been “done” by the development team throughout the sprint.

Overall, the Scrum methodology provides a guideline for studios. It is recommended that the studio makes changes to the framework depending on the teams’ needs (Atlassian, 2023d).

2.5.1 Scrum in Game Development

In game development, it is common to see scrum methodologies being modified to fit the development teams’ needs better.

The classic scrum provides a template of guidelines for software companies to follow. For example, one of the modifications we do at the studio where I work is that we add another topic to discuss to our sprint retrospective, where we additionally go over what went wrong as well. Discussing what went wrong during a sprint adds a lot of value if a feature is crucial for the project’s development, and if so organizing our sprint planning to improve that feature may be very valuable for the end product.

Additionally, a common issue being seen is that the bigger a studio gets, the more time would be spent on a daily stand-up. Fane (2018) discussed her agile journey with the video game studio Bungie (2023) during the development of Destiny 2 (2017). Since, Bungie is a studio with over 900 employees, if everyone one by one talked about their schedule for the day during the daily scrum, it would take a long time. Especially, for that specific scrum meeting, Fane (2018) made modifications that still made it possible to have daily stand-ups in the studio while being split into several small cross-disciplinary teams.

The scrum model suggested by Kristiadi et al. (2019) uses the classic scrum approach with phases. The phases mentioned in this framework are keywords that are related to video game development, which are the Pre-Production (Planning) Phase, Production (Development) Phase, and Post-Production (Closure) Phase, respectively (see Figure 2).
The Pre-Production Phase (Planning) summarises the plan for the game that will be developed in terms of requirement specification, game design document (GDD), and detailed descriptions of the responsibilities for all departments. The requirement specification aids the developers in understanding the user requirements and the features needed to be present in the product backlog. The game design document forms the base for the project that contains everything needed for the goals and the list of objectives during the development. Lastly, the separation of the tasks for each team member. This step is the clarification of what each member will be responsible for in the project which is closely related to the product backlog since each item will be assigned to a team or a member to take on.

The Production (Development) Phase is focused on creating the game by picking a product backlog to become a sprint backlog for the developers to work on during each sprint. The process for the production phase takes four steps to complete. First, the designs of the game elements and their items need to be clarified. Second, the game engine suited for the project should be chosen and the designed elements should be developed accordingly. Third, testing the game in two different ways helps the developers about what works and does not work within the game. Two testing methods mentioned by Kristiadi et al. (2019) are beta testing and heuristic-based testing. Beta testing assists the developers in finding bugs that affect the functioning of the game and heuristic-based testing allows the developers to see how enjoyable and accessible players find the game. The suggested sprint cycle duration is one month, however, there have been increases in productivity when the team determines the sprint duration for themselves.

The Post-Production (Closure) Phase is the phase that takes place after the testing and quality assurance. This stage is focused on creating the final documentation needed for the marketing and making the game into a finished product.

These stages are exemplary, the amount and the focus of them can vary between projects.
2.5.2 Positive Impacts of Scrum Methodology

The results of Koutonen and Leppänen’s (2013) study with the Finnish game studios show that the usage of agile practices in studios increases the communication between professionals, the quality of the games, finding the fun in the game, team awareness and creating new features more quickly.

In addition to development, Koutonen and Leppänen (2013) found that project and scope management got easier with the usage of agile practices. Studios were also more consistent in being able to stick to the set deadlines (Koutonen and Leppänen, 2013).

However, there are still issues in the games industry that the developers go through. Similar to how Jain and Suman (2018) categorize some of the software development challenges, the next section goes through some of the issues found during the development of a game.

2.6 Issues in Game Development

Choosing the right type of methodology plays a crucial role in balancing the developers, time, and cost of the project (Keith, 2010; Kristiadi, et al., 2019). When the chosen methodology does not stick within the team, the development phases can get messy. The findings by Petrillo et al. (2008) show that one of the biggest issues that causes problems in game development is the adaptation of a poor methodology, which leads to projects taking a longer time than it is supposed to.

The most commonly mentioned issue in game development is scope problems (Petrillo et al., 2008; Whitson, 2020). Scope issues arise when the development becomes too big for the team (Whitson, 2020), and when unplanned features get added during the production phase of development, which game developers refer to as feature creep (Petrillo, et al., 2008).

Delivering the game on time, the scheduling of the game can also be problematic in game development (Petrillo et al., 2008; Whitson, 2020). There can be a lot of factors why the development process can slow down (Petrillo, et al., 2008), however, they are usually related to unexpected dependencies or meetings occurring which causes the time estimation of some tasks to be larger than usual.

The developers sometimes have to perform extreme amounts of work, typically before the release of the game, the working hours can go up to more than 12 hours a day and working days to 6 to 7 days a week (Petrillo, et al., 2008). This period is called a crunch in the games industry, and the games industry goes through more crunch periods compared to other software development industries (Petrillo, et al., 2008).

During development, technological problems such as integration of new tools into the game engine, and having the game work on a new platform (before it’s released to the market) as a launch title (Petrillo, et al., 2008). The integrations and the new platform take time, and effort to learn from scratch.

However, Whitson (2020) argues that the solutions to all of these should be looked at in more social situations, rather than technological areas. Throughout the thesis, the focus is on the social aspects of game development by investigating if adapted agile development strategies affect cross-disciplinary communication in any way.
3 Problem

The market for video games is very big and it is still growing (Dataspelsbranschen, 2022). According to the Interactive Software Federations of Europe (ISFE), in 2022 the worth of the European video game industry is 23.3 billion euros (ISFE, 2022). However, even with this growth, according to Engström (2020), the topic of game development in academics appears less than articles discussing gamification, serious games, and educational games.

Previous research related to agile development and game development (Petrillo, et al., 2008; Koutonen and Leppänen, 2013; Kristiadi, et al., 2019) focuses more on issues that are commonly debated about in game developer postmortem. Similar to how Whitson (2020) argues that the underlying solutions to issues encountered in game development are related to social aspects, I was motivated to explore topics regarding team communication in practical game development.

The purpose of this study is to showcase how agile development modifications can affect cross-disciplinary communication in game development. Therefore, the formulated research question is:

How do agile development adaptations affect communications between team members from different disciplines?

With this study, I would like to be able to contribute to academic research within the domain of game development, with the findings of software development strategies affecting cross-disciplinary communication in video game studios.

Similar to previous research (Petrillo, et al., 2008; Koutonen and Leppänen, 2013; Whitson, 2020), the methods used for this study are an analysis of professional developer talks, the field study and the survey conducted with professional game developers.

3.1 Method

Since the research is divided into three parts; the GDC talks, the field study and the survey, there were different methods used for data collection in each section.

3.1.1 Content Analysis

The main method used was content analysis; after presenting the results of what has been obtained from the GDC talks, the field study and the survey, a content analysis was performed to find patterns between each source.

The pattern-finding was focused on only the agile methodology adaptations that impacted cross-disciplinary communication.

To get insights from professional game developers, four Game Developers Conference (GDC) talks were chosen about game development.

GDC (Informa PLC, 2023a) is an annual conference in which professional game developers around the world come together to give talks, attend events and network. The talks given at the conference are all about games and mostly about game development, and discipline-specific topics. In order to find conferences that were relevant to my research area, keywords such as “agile development”, “project frameworks”, and “team communication” have been searched on GDC Vault (Informa PLC, 2023b), on the official website that contains the recordings of the panels.

The result of the search was a total of 16 conference talks. However, not all the talks covered both agile development and cross-disciplinary communication; some were mainly focused on
production and marketing strategies that related to iterative development. A few of the talks were also dated back to 2016 and before, so the focus was shifted towards more recent talks as well.

With the usage of double filtering the GDC talks, four panels were chosen to be analysed: Improving the Culture of Critique: Communicating Across Disciplines by Jeff Hesser (2017), Managing Conflict on Small Teams by Rebekah Saltzman (2017), Epic Sync: Wrangling the Work of Highly Interdependent Dev Teams at Bungie by Linda Fane (2018), and Taming the Chaos: Lessons in Project Management by Clinton Keith (2020). These panels differ in terms of the video game studio size and culture, which allowed me to analyse what kind of miscommunication issues can arise in different game development companies.

3.1.2 Field Study: Participatory and Direct Observation

In addition to GDC talks, a field study was conducted at a video game studio with less than 30 employees and on-site. My discipline in the studio was programming, and my work consisted of mostly programming and some designing, which allowed me to observe the communication between the Programming and the Design disciplines closely. This field study was conducted throughout the first two months of working with the studio.

Having direct access to an organization related to my research, allowed me to process the social context that the developers were in better (Patton, 2002). I was present in the agile development cycle including meetings, daily scrums, sprint retrospectives and reviews.

According to Patton’s (2002) dimensions of fieldwork variations, I acted as a part participant and part observer throughout this field study. I gained an insider perspective on how a studio with less than or equal to thirty employees performs cross-disciplinary communication, what kind of software frameworks are used and their effect on communication across disciplines. Having conducted this fieldwork as a solo researcher, field notes were chosen as the main method to obtain data regarding my research. Utilising field notes additionally helped in making a qualitative analysis for the later sections of the thesis.

A data collection agreement was signed before the field study was conducted to get consent from the studio and to make sure which type of data I was allowed to collect. This agreement clarified that the duration of the fieldwork would be long-term, two months, and the focus of the observation is narrow with only focusing on cross-disciplinary communication and agile methodologies.

This method is divided into two parts, participative observation and direct observation.

During the first two months of work, I have attended several meetings, retrospectives, and daily stand-ups, and participated in discussions with team members from other disciplines. This part of the field study is focused on the interactions that I have been a part of that are related to the research.

Field notes regarding the topics listed below have been analysed:

- Types of meetings I have participated in
- Usage of methodologies outside Scrum
- An overview of a sprint with my responsibilities
- Interaction with other disciplines
- Modifications made on agile development strategies

Another way of assessing the cross-disciplinary interactions was by conducting direct observation throughout the office. The main motivation for using this method was to see which disciplines interact with each other most, in which cases and why, without any outside interruptions (Patton, 2002).
Throughout my work days, observations were made in terms of seeing what type of meetings were being conducted, which disciplines were participating and an overall observation of which disciplines communicated with each other the most.

Furthermore, a survey was designed to get more input from professional game developers about communication across disciplines.

### 3.1.3 Survey

As the last part of the research, a survey was designed to make connections to the data gathered from the GDC talks and the field study. Since surveys are generally related to questionnaires and qualitative studies for making connections to numerical values from the given answers to the forms (Punch, 2003), the main motivation for using a survey for this research is to have a number of game developers' opinions about the communication across disciplines and how it gets affected when adaptations are made to the software development practices in a limited amount of time.

The survey (see Appendix A) was designed for video game developers all around the world. Snowball or chain sampling was used to recruit participants (Patton, 2002), and the survey was shared within selected professional game development communities. The selected communities were game developer communities that I was a part of, which allowed some of the developers to share the survey with other communities or developers.

The survey was designed to be anonymous, and the participants were allowed to resign from the study at any point in time. Participants were able to contact me through my educational email and have their answers removed from the study.

It was split up into three sections, basic information, development methodologies and communications across disciplines.

The basic information section of the survey, collected data about the size of the studio, the work location type and which discipline the participant worked at.

The development methodologies section collected information about what kind of project management methodologies were used in the studio and the participant’s discipline.

Lastly, the communications across disciplines section went through both open-ended and multiple-choice questions about agile methodologies having an impact on cross-disciplinary communication in the studio.

Participants were surveyed using Microsoft Forms ([https://forms.microsoft.com/](https://forms.microsoft.com/)). The software Microsoft Forms was chosen because of its branching questions feature, to make some questions dependent on the previous answers. This ensured that the participants would see all the questions relevant to them and it eliminated the additional captions in the questions asking the participants to leave the answer field blank if they had picked “Multiple Choice A” as the answer in the previous question.

The survey was accessible from the 14th of August 2023 to the 13th of October 2023.
4 Results

In this chapter results of GDC Talks, the field study and the survey will be discussed respectively.

4.1 Game Developer Talks

This part of the results presents the topics the developers discuss throughout their talk which are related to the research question. Panels discussing agile development and cross-disciplinary communication were taken into account, and a total of four talks were picked to conduct a content analysis later on. Previously mentioned in the Problem section of the paper, the panels chosen were: Improving the Culture of Critique: Communicating Across Disciplines by Jeff Hesser (2017), Managing Conflict on Small Teams by Rebekah Saltsman (2017), Epic Sync: Wrangling the Work of Highly Interdependent Dev Teams at Bungie by Linda Fane (2018), and Taming the Chaos: Lessons in Project Management by Clinton Keith (2020).

The selected talks differ in terms of the discipline of the presenter and the video game studio the presenter works at. This enabled me to obtain more data, in terms of studio size, culture and perception of different disciplines, in a limited amount of time.

Table 1 Selected GDC talks, the years of the conference, the presenter and their discipline.

<table>
<thead>
<tr>
<th>GDC Talk</th>
<th>Year</th>
<th>Presenter</th>
<th>Discipline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improving the Culture of Critique: Communicating Across Disciplines</td>
<td>2017</td>
<td>Jeff Hesser</td>
<td>Art</td>
</tr>
<tr>
<td>Managing Conflict on Small Teams</td>
<td>2017</td>
<td>Rebekah Saltsman</td>
<td>Co-Founder, Management</td>
</tr>
<tr>
<td>Epic Sync: Wrangling the Work of Highly Interdependent Dev Teams at Bungie</td>
<td>2018</td>
<td>Linda Fane</td>
<td>Agile Coach, Management</td>
</tr>
<tr>
<td>Taming the Chaos: Lessons in Project Management</td>
<td>2020</td>
<td>Clinton Keith</td>
<td>Agile Coach, Producer</td>
</tr>
</tbody>
</table>

4.1.1 Improving the Culture of Critique: Communicating Across Disciplines by Jeff Hesser

The talk Improving the Culture of Critique: Communicating Across Disciplines by Jeff Hesser (2017) that took place in GDC 2017 is about the issues that can arise or ideas that can be missed out on if developers do not communicate across disciplines. Hesser (2017) is fascinated by how people assess and talk about each other’s work. He gave another talk in 2016 about giving or receiving feedback that enhances the quality of the conversation on an individual level, in this talk (Hesser, 2017) he is focusing on how the culture of critique can be applied in cross-disciplinary communication.

Disciplines in a studio have some overlap with others when it comes to certain topics. These topics are shared across the whole studio; but there are a lot of places where the disciplines do not overlap as well (Hesser, 2017). The disciplines can interpret situations differently than others and have their own technical language when it comes to communicating specifics
(Hesser, 2017). In Figure 3, a graph from Hesser’s presentation (2017) showcases which disciplines the artists at Harmonix (2023) interact with.

**Figure 3** The graph of artists’ connections with other disciplines in the studio (Hesser, 2017).

He mentions that it’s especially hard to communicate about the disciplines when it’s not overlapping with the other disciplines’ works. However, it is also the most vital part of the discussion since every discipline brings in their own ideas about their work (Hesser, 2017).

Hesser (2017) explains the word CRIT, and how it can lead to two understandings, criticism and critique. He interprets the word criticism as a personal judgement, a first reaction response to something; on the other hand, the word critique means something more objective, analytical and reflective approach. For Hesser (2017) a conversation should not be one-sided where there is a giver and a receiver, but more of a two-way conversation where everyone gets involved. The giver needs to think about how they are providing this information, whether is it in a way that the receiver can interpret; and the receiver thinks about how well are they receiving the information and what can they do to improve it (Hesser, 2017).

The developers will need to think about how their work will be seen by others. An example that Hesser (2017) mentions is his time as a Lead Artist in Harmonix (2023) during the development of Fantasia: Music Evolved (2013) when the artists placed out grey boxes throughout the environment, the other developers did not understand what they were looking at (see in Figure 4). With the help of concept artists, the presentation became more clear.
Hesser (2017) points out the importance of in order for critique to be given, the presented material should also be in a state where it can be understood by other disciplines. This leads to his second point of the talk, empathy. He illustrates being empathetic not necessarily being emotional and sympathetic, but more towards being able to think from another one’s perspective. However, this point of view can not represent the actual perspective by allowing ourselves a way where we can imagine how the disciplines would like to be seen as is a mindset that can overall help the communication across disciplines (Hesser, 2017).

To help communication across disciplines Hesser (2017) highlights the key points to starting an empathic conversation; feeling respected, heard, and valued. These points do not mean that the critique is all about being nice to one another, but more towards being direct, open and in some cases harsh while maintaining the core points that were mentioned before (Hesser, 2017).

The remaining part of the talk is about the strategies that can be applied to improve the conversations and Hesser (2017) divides these into two sections; what can be done before the conversation, and what can be done during the conversation.

Before the conversations, Hesser (2017) suggests having kick-off meetings so the team gets to know what everyone is going to be doing and who will they work with throughout the given development time. Hesser (2017) gives another example from the development of Fantasia: Music Evolved (2013) where the concept artists were working closely with narrative designers and the 3D artists were testing out gameplay concepts with game designers. Over time, there was confusion when the created features ended up being independent of one another since there was not a meeting where the planning of the work was discussed beforehand (Hesser, 2017).

Another strategy that may have helped development was meeting regularly on how things are going with the tasks at hand (Hesser, 2017). Having these meetings regularly would ensure that everyone would know what they were working on, it could have acted as a quick update (Hesser, 2017).

To be able to provide critique, Hesser (2017) shares preparational steps to get ready to be able to give critique. First of the step is to get to know what is going to be critiqued. Starting by describing it objectively; what it is, and what is it doing, which in the end gives the developers...
time to not be judgemental about the critique (Hesser, 2017). The second is to describe the subjective reactions towards what is being shown, and how it makes me feel when I look at it.

After getting to know and experience the thing being shown, analysing it comes next (Hesser, 2017). Getting answers to what is the purpose of this thing, what is it trying to achieve. In addition to that, hearing about the maker’s perspective on how they are planning to achieve the goal with it and lastly, analysing how it will achieve the goals that it proposes or falls short (Hesser, 2017). It is important to note that the question becomes how, and where the process is important rather than knowing whether it can achieve it or not.

Another structure that the company uses is to gather developers across disciplines to have brainstorming sessions to solve problems (Hesser, 2017). Everyone from different disciplines brings in ideas, addressing their concerns from their perspectives which forms a decision to be made in the end about the issue being discussed.

Before having a conversation, a way that can help the conversation is to research the problem ahead of time (Hesser, 2017). If there is a question that a developer would like to ask someone, what is suggested by Hesser (2017) is for them to research what the solution might be and grasp the problem on their own for some time before immediately trying to get an answer. This small search might help them find the solution quickly but can also help in understanding the terms the other developer is talking about since some research has been done before.

Preparing quick prototypes to showcase your idea can help the team to see if it is feasible to iterate over it (Hesser, 2017). It is an easy way to show if the idea can lead to something positive for the game that is being built.

Even before making a prototype inside the game engine, showcasing an idea with a paper prototype by explaining how the idea is going to work out to others. The team can iterate on what has been shown over time and make progress with other types of prototyping, such as digital or in-game prototypes to see how the idea feels on a player experience level.

Prior to having a critique session, it is important to define the problem that the team is trying to solve in detail (Hesser, 2017). These problems are not specifically bugs, but more towards the feature in the game not giving the right feeling towards the developers, making the developers feel that the feature itself is not working inside the game (Hesser, 2017). When a problem is clearly addressed solving it becomes much easier during the critique.

The last strategy Hesser (2017) talks about preparation before having a conversation is building relationships. A simple act of sharing appreciation and respect helps the team get motivated and feel heard. For this part Hesser (2017) only gives a two-step advice; if you see something, say something. Saying to one another that you like what they have been doing means a lot.

Building relationships through asking for help is another step Hesser (2017) talks about. In one project that he was working on, he was making animations and had trouble adjusting the animations if the hierarchy of the game objects was changed. Once he asked one of the engineers in the team, he learned that there was a tool that they had developed to fix that issue from breaking the animation in the game (Hesser, 2017). He did not know about the tool beforehand, and he did not reach out to other disciplines about it before. Asking for help from other disciplines can reduce the time you spend working on the problem by yourself, and it is another way to build a relationship by helping each other (Hesser, 2017).

Having organized events in the team is additionally a great way to build relationships, to get together as a team outside of work. It helps every one to get to know one another and socialize, especially people with whom you do not get to talk at work as often (Hesser, 2017).
Hesser (2017) moves on to the second part of the strategies, which are things we can do while having a conversation, that involves critical feedback involving other disciplines. He starts by raising the question of, what goes well during a conversation. Hesser (2017) suggests moderators can help to facilitate empathic communication by, focusing on the critique, clarifying communication, and seeing the conversations from outside. However, this saying does not mean that they have hired moderators into the company, it means that anyone who is having a conversation can be a moderator (Hesser, 2017).

Giving your perspective on things that you have been working on is important for anyone listening in on the conversation to understand where you are coming to this conclusion (Hesser, 2017). This can also lead to the developers asking other disciplines what they think about it as a programmer or a designer. Getting input from how the other disciplines see things helps in getting more advice about the issues. This establishes the values of all points of view during the conversation (Hesser, 2017).

Reflecting on the ideas being exchanged during the conversation helps the team come to the same page when discussing the issue (Hesser, 2017). Especially while working across disciplines, Hesser (2017) values people having the translation technique to understand how different disciplines often talk about the same thing but with different wordings. This also shows that you are paying attention, you are listening, and caring about the topic being discussed and helps your feedback to be heard as well (Hesser, 2017).

Asking questions when giving feedback is another strategy that can be used during the conversation (Hesser, 2017). Asking questions usually means that the assumption we are making is wrong (Hesser, 2017). However, it can save time and energy, rather than coming up to another developer and complaining about the thing that is not working, you ask questions on what the problem could be to figure out solutions. Similar to the other strategies, this also makes the developers feel included (Hesser, 2017).

Asking questions when receiving feedback is the final strategy that Hesser (2017) suggests to developers. In a situation where the feedback may not have been phrased well, pursuing the raised issue by asking questions to figure out what is trying to mean clarifies the problem and allows the developer to work on the solution better. Since Hesser (2017) works as an artist at Harmonix (2023), he has experienced that non-artists find it hard to talk about art. This can be applied to other disciplines; however, it does not mean that they do not have good opinions about it.

These are strategies that Hesser (2017) emphasises during his talk, but sometimes things do not work out the way they are and people can disagree. Having a structure in the company can help the developers deal with the situation whenever it occurs. The strategies mentioned by Hesser (2017) help the developers to be able to disagree with respect and be understanding of one another. When emotional relationships break, they are stronger when healed (Hesser, 2017); getting through disagreements can improve the relationship.

As the final message, Hesser (2017) highlights the importance of behaving with other people’s experiences in mind and focusing on empathic communication.

4.1.2 Managing Conflict on Small Teams by Rebekah Saltsman

In Saltsman’s panel (2017), Managing Conflict on Small Teams, at GDC 2017, the topic of avoiding toxicity by maintaining good conflict at Finji (2023). The talk revolves around the problems that have been occurring in the development team, and how Finji (2023) tackled the stated issues. The team has experienced how the wrong type of conflicts can lead to chaotic outcomes; such as cancelled games, friendships lost, sleepless nights, and money wasted (Saltsman, 2017). Having experienced these outcomes Saltsman (2017) has come up with solutions to lower the risk of these catastrophic results.
1. Using a planning and design process that emphasizes solutions and downplays ego.
2. Maintain flat and concrete discussions of plans and design merits.
3. Learn how to postpone contentious decisions as long as it is not a bottleneck to current development.
4. Trust but verify your experts within each discipline.
5. Always engage external mentors, especially for long-term planning.

To find a solution to a problem, firstly the problem itself should be identified. Referring to the first suggested bullet point by Saltsman (2017), during the development phase developers can find themselves not liking a specific part of the game, feeling unsure about a build, or having trouble with the player experience; which leads to them complaining about the game however no solid identification of a problem is mentioned. Saltsman (2017) says the problem itself must be identified in its detail.

Whoever in the team identifies the problem, puts in the effort of coming up with a solution as well (Saltsman, 2017), so the other team members can see where the solution is coming from and why this is the solution (Saltsman, 2017).

Lastly, the idea explaining the steps to solve the problem gets presented to the team. The discussants go through the positives and the negatives of the solution they found. Positives can be the fix to the problem or something new that can make a difference in the project. Negatives can be the time it takes to develop the solution, changes to the infrastructure, having the possibility to affect other systems or assets in the game (Saltsman, 2017). After having mentioned the positives and the risks, the team gets to discuss the idea and everyone in the team is expected to show open-mindedness and mutual respect to others (Saltsman, 2017).

However, these steps are not a written rulebook at Finji (2023), they merely represent the guidelines at Finji (2023), which additionally showcases that it is a collaborative process within the team (Saltsman, 2017). Saltsman (2017) mentions that the reason these steps have been working for the team is that from the start of the development, everyone in the team shares the same goal: “The game matters more than our egos.”. The tone, and target audience for the game are decided in the early stages of the development, if the discussions are not complimenting the style then it might be the case that the developers are building a different game.

To test out if an idea works or not, concrete prototypes that demonstrate what it can be in the game back up the viability of the idea. The team treats game design similarly to science, making claims that can both be proven right and wrong. It is important that the team sticks to solving the problem and not get carried away by only theoretical discussions that can be interesting but not necessarily help out the idea.

Problems are tackled in this way at Finji because everyone in the team gets to see what the new features will be, has a chance to contribute to the design of the game, and can provide feedback before fully committing to the implementation (Saltsman, 2017).

During development, there may be moments when the team can not agree on a specific solution, where the issue is contentious. This is the point where postponing the decision until more information arises can be a temporary solution, deciding to move on without implementing anything new since the development of the game can continue without it (Saltsman, 2017). However, before postponing the team needs to make sure that it is not bottlenecking the current development and that there is a budget to be able to postpone this feature.

Saltsman (2017) notes that iterative game development is usually the case with indie game companies. Ideas are flowing as the game’s development progresses, but this may not be the case with bigger companies when there is solid documentation that the development team
follows. With each passing day, more ideas flow into the development that can provide solutions to the occurring problems.

Another approach Saltsman (2017) talks about is the “intermediate impossibles”. This is when postponing a feature is not possible and the team has to come up with a fast solution to be able to continue development. These elements can be placeholders to be replaced sooner or later in development. During the development of Overland (2019), the team brought the placeholder enemy avatars into the 3D space which did not match the overall theme of the game (see Figure 5). However, they did not know what the enemies were going to look like so they iterated over it again and again until the design felt right.

**Figure 5** Placeholder pink pixelated enemies in the 3D space (Saltsman, 2017).

If none of the suggested guidelines worked for the issue, the last step to take is to leave the decision-making to the lead developers. Saltsman (2017) emphasizes trusting the team and especially the lead developers to make the right call when in need. This does not mean that the most experienced person in the room will always make the right call, Saltsman (2017) says “I do not care how many games you have shipped, I do not care about your ego, I do not care how long you have been in the industry, all I care about is that you have the same shared goal.”, she emphasizes the trust in the whole team and not the hierarchy.

Trusting the lead developers with a decision still comes with verifying the strengths and the weaknesses of the idea as a group as mentioned before (Saltsman, 2017).

The final statement Saltsman (2017) makes is the importance of opening up to outside sources, and mentors. Testing and giving feedback on the prototypes of the game, going over the business plans of the studio and providing guidance when needed; mentors can provide insights on a project to advise on developing for a target audience and how the industry is looking now for that project. Saltsman (2017) values mentors highly since they help the team realise what is working for the industry and what is not. Paying attention to the mentors can save time and money that would be spent on a failing project.

Additionally, outside sources can help the studio in forming meaningful partnerships, making the studio function better by trying out different practices.
4.1.3 Epic Sync: Wrangling the Work of Highly Interdependent Dev Teams at Bungie by Linda Fane

In 2018 at GDC, Linda Fane, agility coach at Bungie, gave a talk about how she helped Bungie to collaborate more effectively during development. She was also responsible for switching Bungie’s workflow to agile methodology.

Bungie was started as a team where the structure was mostly organised by different disciplines. They were carrying out a traditional waterfall methodology as the tasks were moving one by one between disciplines. The development of a project would start with Art, move onto Animation, and then Design, Engineering and lastly Testing.

This workflow was carried out for several projects that Bungie did, however as their title Destiny (2014) was shipped, one of the producers addressed their concern on how difficult doing anything cross-disciplinary was.

During the development of the second title for the Destiny game series, Destiny 2 (2017) they modified the team structure by creating small cross-disciplinary teams. These small teams consisted of designers, engineers, artists, and testers; working with agile and scrum methodology. For Bungie, the team size’s limit was nine people. Having ten or more people in a team resulted in communication being much harder to manage. The focus was shifted towards a unified player experience, rather than each discipline focusing on their own individual tasks. Not only how each developer was doing for their tasks, but how that task ended up contributing to the player experience in the end. The disciplines being split up into different small teams, allowed developers to get to know each other’s workflow and adapt to and trust one another. For Linda Fane (2018), cross-discipline teamwork sparks the creativity.

![Figure 6](image)

**Figure 6** The representation of the cross-disciplinary team at Bungie (Fane, 2018). The colours represent different disciplines in the studio; blue: Art, green: Engineering, yellow: Animation, orange: Design, purple: Testing.

Bungie was able to form more than forty small cross-disciplinary teams with a different feature focus for Destiny 2 (2017) (see Figure 6). This was the first step of adjusting the company to a new workflow; the second step was to make sure that the teams were synchronised. These teams are working towards the same goal; to make a unified game however, they are focusing on different parts that make the game a whole and the studio has to keep all the teams on track.
As seen in Figure 7, several different teams worked together on different parts of the game. Bungie was split into small teams of clearly focused tasks, however, there were systems in the game that needed the work of a couple of the small teams together. This resulted in dependencies being formed between the small teams.

The agile manifesto (Beck, et al., 2001) suggests that an agile scrum team does not have any dependencies, however in Fane’s (2018) panel, the problem at hand was sorting out how to get rid of the dependencies the content development teams had because there was not an on-paper solution to solving dependencies in big teams.

All the teams knew the goal that they were trying to achieve, however, they lacked the communication and coordination to stay aligned with each team’s progress and goals. After each sprint, the teams would drift away with time. To avoid the progress flow from shifting, Fane (2018) refers to the first value and the sixth principle in the agile manifesto (Beck, et al., 2001). The first value is individuals and interactions over processes and tools, and the sixth principle is that the most efficient and effective method of conveying information to and within a development team is face-to-face conversation. She emphasizes that face-to-face communication helps a lot in getting the point across, so they want to be able to provide a way of direct communication for the teams to get in sync.

To get every team in sync, Fane (2018) added a cross-team ritual, which was a planning day where all the development teams came together in a big conference room to plan the next sprint and discuss the dependencies. It was crucial that this planning day was done offsite to ensure everyone would have the opportunity to do face-to-face communication. During the cross-team ritual, all the developers from all the teams gather up in a room and discuss the progress with each other, explaining the blockers and questions they have for the other teams.

For the dependencies to be resolved, each team has its board which contains the planned tasks for the quarter, at Bungie this is translated into 6 sprints (Fane, 2018), and the dependencies they have regarding the other teams, as well as what the team is committing this sprint, what is expected to be completed and lastly the risks and the open questions sections the team is providing.
In the conference room, there is a big dependency board with the team names as columns and sprints as rows. There is also a risk board, the developers put up cards if the issue mentioned can not be resolved with discussions.

Figure 8  The planning day responsibilities for each team (Fane, 2018).

On this planning day, since the work being done is only planning, all the teams have different roles and responsibilities that they must perform for the day (see Figure 8). The development team gets to discuss the team’s work with the product owner. The product owner reviews the scope of what the team can do and handles if that specific task should be done by the team.

On the other hand, the scrum masters, which are usually the producers at Bungie, discuss the feasibility of the given tasks to the team. Some other special tasks for the day are putting up post-its on other teams’ boards if they have a dependency in the other team and organizational tasks such as categorizing the post-its on the team’s board and additionally transferring the notes made into the electronic system where the backlog is located. Just for this gathering, there are new roles formed and responsibilities throughout the day.

Apart from the content development teams, other roles support the whole planning phase to have the event fit in a day. The directors and the product owners, help the teams to structure the main pillars that they present at the beginning of the day to keep the team’s work aligned with the goals with reachable goals. The team release leadership consists of the leads from all the disciplines gives directions, and makes decisions to resolve the blockage caused by the dependencies in the content teams.

By adding a day into the development phase, Bungie was able to keep track of how each team was progressing after six sprints and make sure the dependencies were resolved faster than before. Fane says “One team reported that there was an issue that had been unresolved and lingering for two months, and with the right people in the room on planning day, it was resolved in ten minutes.”

Lastly, Fane herself was a part of the supporting actors during this day. She had the responsibility of managing the dependency board. She keeps track of all the dependencies that have been posted on the board that day and makes sure that the developers solve the mentioned issues before the day ends.

Fane (2018) presents some examples of how teams can come to conclusions when it comes to dependencies in tasks. However, this is not the case every time. If the teams can not solve the dependencies, the dependency becomes a risk and it gets into the hands of the directors and
the release team. It becomes the director's and the release team’s responsibility to discuss it with the two teams that have presented the issue to find a solution.

**Figure 9** Different variations of planning day at Bungie (Fane, 2018).

For one year and a half, during the production of Destiny 2, Bungie experimented with different variations of the planning day (see Figure 9) (Fane, 2018). This has made them easier to adjust depending on the project team size and even the clients that they were not able to meet on-site which required them to try out virtual planning days.

The planning day overall had a massive positive impact on the predictability and agility of the company. The content teams got aligned and updated on each other’s work, every six sprints all of the team members were aware of what others were planning on working on for the next sprint and did dependency checks before the tasks were decided on. Team discussions were made face-to-face, allowing the topics to be talked about faster and more directly. Which resulted in an efficient day that allowed all the teams to be aligned and synced with each other.

Linda Fane’s talk (2018) showcases how modifying the classic scrum model to adapt to bigger teams helps them stay in touch about their progress during development. Fane (2018) emphasizes the importance of forming conversations about the planned tasks if they are creating any dependencies for other teams, is more convenient and direct when everyone is on-site and has the same goals for the day.

**4.1.4 Taming the Chaos: Lessons in Project Management by Clinton Keith**

In GDC 2020, Clinton Keith (2020) talks about taming the chaos, the situations where the game development is in disorder and confusion and how can the developers take control of it. Not to be misunderstood, chaos is needed during game development to explore different ideas and find the fun in the game, however, taming the chaos is needed to avoid developers burning out, crunching, making unnecessary features and producing a bad game in the end (Keith, 2020).

Keith (2020) saw a lot of the developers, including himself experience burnout. As a project manager, in three months of crunch, he has turned into a risk avoider from an enthusiastic game developer. Overworking reduced the productivity of the developers and the overall passion that was present at the beginning of the project (Keith, 2020).

The issues that led to the burnout were mostly putting the focus on getting as many tasks as possible done and rejecting ideas that might have provided more fun in the game (Keith, 2020).
To explain the solutions Keith (2020) found to tame the chaos, he took inspiration from the three different systems in the book Antifragile: Things that Gain from Disorder by Nassim Nicholas Taleb. These systems are:

1. Fragile System
2. Robust System
3. Antifragile System

Keith (2020) provides examples of these systems and what they could correspond to in game development.

He describes a fragile system, as a system that breaks under stress. A system that is full of bugs usually works but is very fragile to any outside factors (Keith, 2020).

The robust system is indicated as a system that bears the stress but does not break. This system is very fixed and it makes things very difficult to change or adapt to new ideas (Keith, 2020).

The third system is the antifragile system. This system not only handles the stress but grows under it. The antifragile system brings beneficial stressors to play (Keith, 2020).

Beneficial stressors, allow game developers to think ahead in development. For example, when it comes to bug fixing, with an antifragile system the outlook is to find answers to the questions:

- How and why did that bug into our game?
- How can we prevent that bug from appearing in our game in the future?

Finding solutions to these questions helps developers to encounter fewer problems in the future.

Keith (2020) presents four ideas to tame the chaos that can occur during game development:

1. Focusing on outcomes over output
2. Embracing the risk
3. Managing debt
4. Building a culture where developers thrive

For the first idea, focusing on outcomes over output, Keith (2020) gives an example of the Nintendo method. While he was working with Nintendo of Japan (2023), the development team would have three months of money to find the fun in the game. In the case that the team could not find the fun, the project would get cancelled. Having a three-month deadline caused chaos to appear early on during development.
Applying the Nintendo method caused chaos to appear in the early stages of development (Keith, 2020).

There was one rule to be followed for this execution to work, and that was getting the basic character controls and the cameras set before any levels were implemented (Keith, 2020). Keith (2020) broke down the three-month period into three parts, exploration, emergence and exploitation, respectively.

During the exploration phase, ideas would flow around and there was a focus on finding the main mechanic. Once the idea was found, features needed to test them out would get into development during the emergence phase. Lastly, once the mechanics were built, the team would exploit them throughout the game. Some small irrelevant details in the game might make players have fun more, which can surprise developers, but also allow them to change up the game to how the players would like it with minimal effort (Keith, 2020).

In waterfall development, there is a strict schedule that needs to be followed. These steps are; design, pre-production, production, and alpha/beta stage, respectively. A lot of documentation time is spent throughout development and they cannot be re-arranged or adapted to change during the stages. The main difference between the waterfall and the Nintendo method is the part where the developers focus on finding the fun. In the waterfall method, finding the fun in a game is thrown at the latest stage of the development, which results in sudden changes being made right before the game is released which leads to developers being overworked and falling into crunch.
The developers usually will not be able to get a chance to add the “fun” into the game in the later stages of waterfall since the rest of the time will be focused on bug fixing and polishing.

This rush of development makes it likely to cause conflicts to arise between investors, marketing teams and developers when the chaos occurs in the later stages (Keith, 2020). The work that has been put into the game might face changes, and in some cases get completely removed from the game. There may be new requests coming in from the stakeholders to change or add new features to the game to make it fun. To be able to complete these requests the developers face a time of overtime, in other words, a crunch (Keith, 2020).

Getting a good relationship with the stakeholders early on can help avoid the chaos the team goes through at the end of development (Keith, 2020). Allowing outside sources to provide feedback and test the builds of the game helps the development to get on the right track and helps the development team to build trust with the investors. Another strategy Keith (202) advises is building a shippable quality of the game every three months, which helps the producers to make new or adjust the current development plans for the upcoming period.

However, due to other factors, the plan may not always allow the developers to use the Nintendo method (Keith, 2020). Depending on the studio culture, detailed documentation might be still valued more than iterative development which would take the time off from finding the fun (Keith, 2020).

The second idea that Keith (2020) talks about is embracing risks. In the case of a risk occurring during the development, he suggests the developers be proactive about it. Deriving on that, Keith (2020) gives a flow of risk management (see Figure 12).
He explains these steps by going over a scenario where the identified risk is missing the critical launch date of a project. After identifying the problem, the next suggested step is to find the root cause that made the problem occur in the first place, in the example, the cause is a key middleware not being ported to a platform. The next step is to find whether this issue is going to come true or not by setting a deadline for the porting of the middleware to the specified platform. If this porting does not happen, then there is a risk for the issue to occur, which leads to the last step in the cycle, finding a solution to mitigate the risk. However, the risk management does not end there, there is a need to repeat over if there is another deadline to be set to check for possible arising risks.

This risk management strategy may not look good for outside sources, since most of the time they would like to hear about the good news and not the risks that can come up (Keith, 2020). However, coming up with a mitigation plan before taking the risks with the stakeholders, can help the outside sources be more calm about the situation (Keith, 2020). As the development team, showing that there is a plan in case of a risk, will also show the stakeholders that there is preparation and that the team is open about the progress of the development (Keith, 2020). Repeating risk mitigation on a regular basis, the suggested time cycle by Keith (2020) is every six to twelve weeks, which helps the team to reduce the risks as much as possible.

The third idea to tame the chaos is managing debt (Keith, 2020). Keith (2020) mentions different types of debt that can occur during development:

1. Technical
2. Design
3. Content
4. Production
5. Shipping

All the debts that can pile up, affect the shipping date of the game (Keith, 2020).

A lot of agile practices check up on the features in the game and check the definition of done (Keith, 2020). The definition of done can vary between studios, from is this feature causing a bug or does it passes all the automated tests (Keith, 2020).
As seen in Figure 13, Keith (2020) shows how the increase in debt affects the shipping debt over time which forces the developers to make compromises, in the end, to make the game shippable.

To reduce the shipping debt, Keith (2020) suggests the developers do a *magazine demo*, a polished build of the game, every three months (see Figure 14). Having a stable build allows the developers to have some time to polish the game, even more, to increase the shippable quality of the game in the end.

Iterating this strategy, studios can adjust the repeating period of making a stable build every two weeks, or even every two days, to reduce their shipping debts as much as possible (Keith, 2020). This strategy also goes in parallel with the idea of continuously improving, iterating and adapting.

Depending on the studio’s budget, doing automated testing overnight can help the developers in finding bugs, or small issues where an object has been moved that caused a blockage in a level (Keith, 2020). This ensures that the developers would not have to spend time at work testing the level, since the automation can be done after work hours (Keith, 2020).

Fixing stable builds from time to time reduces the shipping debt but stakeholders might prefer the team working on new features rather than bugs which can affect the time to do a *magazine*
Moving the debt reduction into the development plan may make the developers feel more comfortable about the new arrangements, and stakeholders may be more open about it once the results of the builds improve as well (Keith, 2020).

The last idea Keith (2020) talks about maintaining chaos is about building a culture where developers flourish. He emphasizes that this is one of the most productive and important parts of dealing with chaos.

Keith (2020), as a manager realised that working overtime in a studio starts well in terms of getting more work done, however, as time passed the scope went under the normal amount of work that was being done without any crunch. The more time passed, the more tired the developers got and they started to work more slowly (Keith, 2020).

Keith (2020) says “Instead of focusing on vanity metrics, focus on accountability.”, instead of focusing on how many hours the developers worked in a day, measure the productivity by how much value has been added to the game.

When people were working overtime, and the management asked them to stay late to work, it represented a lack of trust in the team (Keith, 2020). In regards to this, Keith (2020) presents a graph that shows the five dysfunctions in a team (see Figure 15).

![Pyramid Graph](Image)

**Figure 15** A pyramid graph showcasing the five dysfunctions in a team (Keith, 2020).

The basis for these dysfunctionalities is the absence of trust in the team. The system makes the developers feel that they have accomplished something, only when they complete the tasks, they have for themselves. It shows that there is a clear lack of trust in the developer to create the tasks for them to work on (Keith, 2020).

The untrustworthy behaviour causes a loop where the developers are only focused on the tasks at hand and do not spend time on anything else (Keith, 2020). The reduced trust and collaboration affect the results in the end poorly (see Figure 16).
By gaining trust (see Figure 17), the developers can decide how they can organise their tasks, their responsibilities, and their estimates (Keith, 2020). Practices such as sprints in agile development come into place in these situations and the developer gets to decide their individual tasks and whether they can be done or not. Which shifts the focus from how many tasks have been completed or lines of code have been written to what has been added to the game right now.

There are times, when developers are very motivated and excited to implement new ideas and take on the challenges that they were given to do, without realising that time, is passing since they get very focused and have a connection with the state of flow (Keith, 2020). Arranging the tasks to developers that will allow them to be in the state of flow yields more positive results since the task is neither too difficult to accomplish nor too easy to get bored (Keith, 2020).
Building trust with the developers, early on, is essential to allow the trust cycle to work (Keith, 2020). Having gone through a crunch or a burnout situation with the team might affect how many of the developers might trust the manager again (Keith, 2020).

### 4.2 Field Study Results

Communication between disciplines is important to keep the development progress known by everyone in the studio. Professional game developers and agile coaches have mentioned the importance of open communication, and establishing trust and respect in the team, in the previous section.

Having started to work at a video game company, I have had the chance to observe and participate in several occasions where team communication was the key to solving problems, finding ideas and getting creative.

The field study was conducted on-site at a video game studio with less than or equal to thirty employees. As a front-end-focused programmer, my work mostly consists of designing the user interface (UI) in the game engine and writing code to give functionality.

Throughout two months of work time, I have observed how modifications made in development strategies affect cross-disciplinary communication throughout my daily tasks, meetings, daily stand-ups, and sprint retrospectives and reviews.

#### 4.2.1 Participatory Observation

For the participatory part of the field study; data regarding agile development strategies, a broad overview of my sprint responsibilities, and occurrences of cross-disciplinary communication were noted down.

Throughout these two months, I have been a part of four two-week sprints and have attended several Scrum meetings. As seen in Table 2, the type of meetings and how frequently they occur are specified.

<table>
<thead>
<tr>
<th>Table 2 Type of Scrum meetings and their occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type of Scrum Meeting</strong></td>
</tr>
<tr>
<td>Daily Stand-ups</td>
</tr>
<tr>
<td>Sprint Review</td>
</tr>
<tr>
<td>Sprint Retrospective</td>
</tr>
<tr>
<td>Sprint Planning</td>
</tr>
</tbody>
</table>

Since the studio size is less than or equal to thirty employees, all team members from all disciplines attend the meetings mentioned in Table 2.

During daily stand-ups, each developer takes turns in telling:

- What have I done yesterday?
- What am I going to do today?
- Mentions any questions or blockers

Through the usage of daily stand-ups, every developer in the studio knows what the tasks of each team member will do that day. Additionally, anyone who asks a question or has blockers, gets a response during the daily scrum or after the stand-up some of the developers gather up to find a solution to the blocker.
At the end of each sprint cycle, a day was dedicated to performing the sprint review, sprint retrospective and sprint planning, respectively.

For the first two sprints, the sprint reviews were in the format of presentations. Each discipline would add what they have done throughout the sprint as presentation slides. During the sprint review, everyone from all the disciplines would show and tell what they have been working on and go through the questions asked throughout the session. However, in the last two sprints, the structure of the review was shifted from going over the presentation to playing the end-of-sprint build of the game. With this structural change, each discipline would still go over the work they have done, but by showing it in the build. Similar to the presentation layout discussions about the work would follow after the work has been shown in the build.

After each sprint review, we would move on to doing the sprint retrospective. The sprint retrospective allowed the development team to reflect on:

- What went well throughout the sprint?
- What went wrong throughout the sprint?
- What can be done differently for the upcoming sprint?

Before doing the sprint planning, we would go over the sprint retrospective and discuss the reflections.

Lastly, we would do sprint planning for the upcoming sprint. Each discipline would form its own groups and create tasks for the assigned sprint backlog items. Every task would be given an effort point, estimating the load of that specific task. After all the tasks have been created, the sprint day would be complete and the development team would be ready to start a new sprint.

During a sprint, my tasks would usually overlap with Design. A broad overview of how one of my sprints went can be seen in Figure 18.

The first step of the sprint is usually spent with Design, going over the game design documents and wireframes for the feature to be implemented.

![Figure 18](image)

**Figure 18** An overview of a sprint cycle for my tasks.

The second step, programming the feature would be the main task of the sprint. Throughout the sprint, for questions and blockers, I mainly got in touch with Programming. However, there were instances where checking in with Design to get feedback on how the feature was progressing was helpful in quickly making adjustments and updating the feature throughout the sprint. Additionally, there have been instances where Art, Audio and Narrative disciplines were contacted throughout the sprints.

Additionally, there were instances where check-ins with the disciplines of Design and Programming were scheduled during the sprint planning meeting. In general, the meetings would be with Design and Programming, however, there were times when Art was also involved.
In some cases, Art and Audio would work alongside the tasks that I was responsible for. The dashed lines shown in Figure 18, represent some of the sprints that I had the chance to work with Art and Audio to add new assets to the feature. Nevertheless, most of my sprints consisted of communicating mainly with Programming and Design.

Throughout the sprint, thanks to the open-plan layout of the office where all the developers share the same floor, it was easy to get in touch with different disciplines. To check in with other disciplines, I would stand up from my desk and walk over to the workstation of my colleagues to discuss any questions I have, feedback, and so on.

Apart from the usage of Scrum methodologies, pair programming, a technique used in the Extreme Programming (XP) framework, was common in the first two sprints. Pair programming is a method where two developers team up on one workstation. Typically, one programmer writes code and the other programmer is the observer. Throughout those two sprints, I was paired up with different programmers to see which features were being worked on which allowed me to see the patterns and structures used in the code. This technique was very helpful in getting to know the project and the systems that exist.

4.2.2 Direct Observation
For the direct observation part of the field study, field notes were taken regarding cross-disciplinary communication besides my participation.

From the field notes, it can be derived that the sprint backlog planning meetings were conducted with the lead developers from different disciplines and the managerial disciplines; usually a day before the sprint review and retrospective. The purpose of the meetings was to decide on which features the development team would focus on for the upcoming sprint. After sprint reviews and retrospectives, the development team gets presented with the decided features, as mentioned previously, the disciplines usually group up and create their tasks for the sprint on the task board.

It was a common sight to see cross-disciplinary communication between every discipline. During the four sprints, every discipline communicated with each other at least once. However, the table does not indicate how many times these disciplines have come together or if the interaction happened in all of the four sprints.

4.3 Survey Results
The results of the survey are presented in two parts. The first section goes over the feedback from the pilot study and how the survey was finalized after that. The second part goes over the data collected from the different sections of the survey, basic information, development methodologies, and communications across disciplines, respectively.

4.3.1 Pilot Study Feedback
The pilot study of the prepared survey provided feedback in terms of clarifying the description of the study (see Appendix A). This version of the survey was shared with a close group of game developers, for its quick and time-saving purpose convenience sampling was used throughout the pilot study (Patton, 2002).

This version of the survey used the word departments instead of disciplines. One pilot study participant got confused about what the keyword departments meant in the description of the survey. Departments in game development can differ with the studio size, consequently, the keyword was changed to disciplines in the final version of the survey. To ensure no other confusion for future participants, examples were provided in the description so that the participants would answer the questions the desired way.
Another part that was strengthened was to give examples of what counts as modifications for agile development. Since my topic is to focus on any modification that makes a difference in communication between different disciplines, providing some traditional examples of agile development in the sub-text field part for the questions aided the participants in being sure of what they changed in their software framework.

### 4.3.2 Basic Information Results

The purpose of the survey was to have professional game developers’ input about whether agile methodologies are affecting cross-disciplinary communications or not. Having reached out to different game development communities, with snowball sampling, a total of five participants filled out the survey.

For the basic information part of the survey, the size of the studio the participants work at, their work location type and the disciplines they are a part of were recorded.

The studio sizes varied between a couple of participants (see Table 3).

<table>
<thead>
<tr>
<th>Size of the Studio</th>
<th>1</th>
<th>≤10</th>
<th>≤30</th>
<th>≤50</th>
<th>≤100</th>
<th>≤250</th>
<th>≤500</th>
<th>≤1,000</th>
<th>≤10,000</th>
<th>&gt; 10,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

As for participants' work location type, three of the participants work hybrid, one participant works fully remote and one participant works on-site.

Each participant was a part of a different discipline in their studio. The responses were as follows:

- UI/UX Design
- Technical Art
- Design
- Animation
- Programming

### 4.3.3 Development Methodologies Results

For the second section of the survey, participants were asked to answer what kind of development strategies they use in both the studio and in their department. Throughout this section, the participants were allowed to write other types of methodologies, however, none of the participants mentioned any methodologies apart from the ones provided to them.

<table>
<thead>
<tr>
<th>Development Methodologies</th>
<th>Agile</th>
<th>Scrum</th>
<th>Waterfall</th>
<th>Kanban</th>
<th>DevOps</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 6 Percentage of development methodologies used in the participants’ discipline

<table>
<thead>
<tr>
<th></th>
<th>Agile</th>
<th>Scrum</th>
<th>Waterfall</th>
<th>Kanban</th>
<th>DevOps</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The results in Tables 6 and 7 showcase that the usage of Agile and Scrum methodologies is very common both studio-wide and discipline-wide. Some of the other traditional software methodologies, such as waterfall, do not yield the same usage of iterative development. All of the participants except one, picked the same methodologies for both the studio and for what they use in their discipline.

4.3.4 Communication Across Disciplines

This section of the survey consisted of open-ended questions and two multiple-choice questions, with a focus on communication across disciplines.

The results shown in Table 7 illustrate that the participants’ disciplines usually have contact with Design and Programming, and rarely interact with Audio or Writing.

Table 7 Participants and the disciplines they regularly (+) and rarely (-) get in contact with; cells that are marked with (*) are not mentioned in the answers; cells that are marked with (X) indicate their discipline; cells with double marks (+, -) indicate that there is contact with only some of the specialised disciplines (for design, it can be game design, level design)

<table>
<thead>
<tr>
<th>Participant’s Discipline</th>
<th>Design</th>
<th>Art</th>
<th>Programming</th>
<th>Audio</th>
<th>Writing</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI/UX Design</td>
<td>X</td>
<td>-</td>
<td>+</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Technical Art</td>
<td>+, -</td>
<td>X, -</td>
<td>+</td>
<td>*</td>
<td>*</td>
<td>+</td>
</tr>
<tr>
<td>Design</td>
<td>X</td>
<td>*</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>*</td>
</tr>
<tr>
<td>Animation</td>
<td>+</td>
<td>X, -</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Programming</td>
<td>+</td>
<td>+</td>
<td>X</td>
<td>-</td>
<td>*</td>
<td>+</td>
</tr>
</tbody>
</table>

Some of the answers also suggest that they are in contact with only some of the sub-disciplines and rarely interact with the other sub-disciplines. This case is seen in two of the participants, one participant who works in technical art, and the other participant who works in animation.

The participant with the discipline of technical art mentioned that their work aligns with Game Design for any UI needs for game modes and characters, however, their work is not tied with Level Design, therefore there is rarely any contact with that sub-discipline.

As for the usage of agile methodologies, every participant mentioned that they do daily stand-ups. All the participants agree that daily stand-ups provide information about checking in with the team and giving insight into what other team members will do throughout the day; however, one participant mentioned that their work needs to be documented regularly, therefore a lot of their time is spent on documenting the scrum meetings.
One of the participants who works at a studio with less than or equal to a hundred employees mentioned that there are two types of stand-ups performed in their studio:

We have a weekly standup with all the teams as well as a daily standup for the Art department. It's helpful for keeping in touch with other artists since we are drawn to work more frequently with design rather than art.

In addition to daily scrum, two participants agreed that splitting the goals into sprints helped in seeing the bigger goal and what sort of tasks need to be done or prioritized clearly. One of the participants added that the discussed tasks should be kept on a board to reduce the risk of tasks being forgotten.

Almost every participant shared different ideas on how communication across disciplines can be improved. Two participants mentioned that having good producers and managers helps the communication between different disciplines. One of the participants highlighted that having managers with insight into how each discipline works, and who have some understanding of programming, art, and design workflows benefits the team a lot.

The other three participants touched upon the topic of having syncs across disciplines. One participant mentioned doing “show and tell” during sprint meetings, where each team takes turns to share what they have worked on, the challenges they encountered and the solutions they found, helpful for keeping the other disciplines updated on what the progress has been for that team.

Another participant suggested having more cross-disciplinary syncs to reduce the risk of misunderstanding:

Have more syncs between disciplines. Now we have one between level design and art and programming and design. However, art-design syncs could help avoid misunderstandings.

Having better documentation on what the other disciplines have worked on was noted by one participant. Their comment was regarding providing a coherent experience for the players. An example was given about having a wiki page for characters, hence each discipline would know what the character’s personality is and can be more creative within their work.

When it comes to making modifications to development methodologies, four of the participants answered that their studio modifies the used methodologies; however, only one participant responded that the adaptations make a difference in cross-disciplinary communication.

The given answers suggest that modifications are made to methodologies depending on the team size, the systems used to measure effort points, the number of stages in development (pre-production, production, etc.), and the starting time for the sprints for each discipline.

As for why these adaptations do not make a difference in communication between different disciplines, participants responded with some cross-disciplines not having a lot of dependencies with each other, the game industry being very secretive on how things get done, and how teams work in silos, especially in bigger sized studios.
5 Analysis

This chapter of the thesis focuses on making a content analysis of the results presented in the previous section. This chapter aims to find patterns regarding cross-disciplinary communication, and in the last section of the chapter connections to agile development modifications are made. The structure is divided similarly on how the results are presented; the analysis of GDC talks, the field study and the survey.

5.1 Game Developer Talks Analysis

This section focuses on presenting the commonly stated points by the presenters of the chosen GDC talks, regarding methods that concentrate on communication that occurs during the development process of games.

First, the commonly mentioned method that helps communication is having meetings, where the development team can check in with one another (Hesser, 2017). However, forming those meetings can differ with various studio sizes. With some certainty, it can be said that in bigger teams, the communication flow across disciplines is more critical, there might be different roles in the studio to help the interaction between disciplines. Agile coaches, managerial positions, and lead developers can be a big help in resolving miscommunications between different disciplines (Fane, 2018; Keith, 2020). This makes it more common to see in larger studios where some developers find it hard to talk to other developers outside of their discipline (Hesser, 2017). Meanwhile in smaller teams, since the whole development team can be considered as one feature team, it is common for every discipline to know how the development process is going, since it is easier to have meetings where everyone is present.

Establishing a structure in cases of disagreements can help the developers follow a pattern to deal with the situation whenever it occurs (Hesser, 2017). The strategies mentioned by Hesser (2017) help the developers to be able to disagree with respect and be understanding of one another. The structure of how to deal with conflicts can depend on studio size and culture. For example, in the video game studio Finji, Saltsman (2017) puts her trust in the lead developers to make the right call in cases where a decision about an issue has not been made yet. In a similar context, if a dependency between two feature teams has not been resolved yet, Fane (2018) directs both of the teams to have discussions with the directors. Saltsman (2017) states that in a few cases, delaying a decision can also be possible, as long as the development stages are not affected.

Regarding development methodologies, there are still cases where traditional software methodologies are applied in game development, however, Saltsman (2017), Fane (2018) and Keith (2020) agree that applying the iterative way of development provides flexibility for the developers to experiment in implementing structures that work best for their team. For example, switching from the traditional software methodology, like the waterfall method, to an agile method, allowed developers in Bungie to perform better (Fane, 2018). The usage of iterative development is also common in indie studios since the direction of the project can shift many times during production because many ideas flow as the game progresses (Saltsman, 2017). Having the flexibility to experiment with different methods, can additionally be useful for developers who are struggling to “find the fun” in the game (Keith, 2020).

In addition to the previously mentioned topics, a lot of highlights have been given throughout the talks about forming trust, and respect between developers with empathic communication (Hesser, 2017; Saltsman, 2017; Keith, 2020). The same stance should also be taken with studios that have contact with outside sources. Being able to open up to outside sources, for example, by sharing a shippable quality of the game allows mentors, and investors to provide feedback (Saltsman, 2017; Keith, 2020). Sharing the progress of development with outside sources can lead the development team in forming meaningful partnerships (Saltsman, 2017;
Keith, 2020). Keith (2020) states that “Having good management, and building trust with the developers, are essential to establish trust throughout the work cycle.”.

5.2 Field Study Analysis

This section of the analysis focuses on the results obtained from the field study.

Having participated in the meetings mentioned in Table 2, with some certainty it can be stated that the development team is one big feature team. Since the size of the studio is less than or equal to thirty, every team member from all the disciplines can attend Scrum meetings without the need for planning big sprint days, as Fane (2018) has organised. This highlights the fact that in every one of those meetings, cross-disciplinary communication was present.

In particular, having performed daily scrum every morning with the team, ensured that every day, an instance of communication across disciplines occurred, which additionally correlates with the results of the direct observation that all the disciplines communicated with one another at least once.

During the sprints, having tasks overlapping with other disciplines increased the communication amount with the corresponding disciplines. Starting the sprint by going over the game design documents, and wireframes with Design, had a positive impact on giving insights on how the feature will work and behave, which helped a lot in planning the code of the system. Checking in with Programming to go over questions and blockers that I had, and sharing the progress with Design while developing the feature helped me apply the feedback more easily.

From the results of the participatory observation, a few instances of agile development modifications were seen throughout the two months of development.

The types of meetings listed in Table 2 are artefacts of the Scrum methodology, therefore with some certainty, it can be said that the studio generally applies the Scrum methodology. However, one of the modifications made was the application of one of the techniques outside of Scrum, called pair programming. With the utilisation of pair programming, I had the chance to get to know the project through different programmers and saw different systems of the game and how they were connected.

In addition to the usage of another technique from another methodology, modifications to Scrum methods were also witnessed. In Scrum, task cards are measured with points assigned to them. The higher the point, the higher the effort that will take for the task to be completed. In the studio, the task cards are assigned with effort points but each task card is measured with time spent on the task as well. Which introduces keeping track of another metric throughout the development. However, the addition of time for the tasks next to effort points helped in forming a relationship between the effort point and the time spent on the task which had a positive effect on managing the scope and making better estimations for future planning.

Another modification regarding task cards was making separate task cards for meetings throughout the sprint. This method helped in keeping track of who spent time in meetings, which led to better estimation of future time tracking. Additionally, noting down what is discussed during meetings, who attended and what was decided to be done on the created meeting task card, gave the development team insight into what has been discussed and decided for a feature.

Moreover, the alteration made to the sprint review structure, where the whole team discussed what they have done throughout the sprint while playing the end-of-sprint build of the project, provided an interactive way of showing and telling what tasks have been completed and which part of the game that it is related to.
It can be derived that the biggest factors for the results obtained throughout the field study that allowed cross-disciplinary communication were; studio size allowing for the same floor to be shared with the whole development team which made face-to-face communication easier, and having everyone from all the disciplines present in daily stand-ups, sprint reviews, and retrospectives.

5.3 Survey Analysis

This part of the analysis focuses on the results obtained from the survey.

Three participants of the survey, work in a studio with a size of less than or equal to fifty employees. Other participants work at bigger companies with sizes less than or equal to hundred and over ten thousand employees. Nevertheless, the development methodologies used throughout the studios are very similar to each other (see Table 5). Even though almost all the participants are from different disciplines, the used methodologies are similar, regardless of which discipline they are from (see Table 6).

A high percentage of the results showcase that agile methodologies are commonly used rather than traditional software methodologies like the waterfall model. This result additionally complements the results found in previous research, where iterative methodologies were preferred over traditional methodologies (Kristiadi et al., 2019).

There is also a very high correlation between the strategies used throughout the studio and what the disciplines in the studio use on their own (see Tables 6 and 7). Only in one of the participants’ answers, the waterfall methodology is being used studio-wide but not throughout their discipline. Previous research also shows that even though there is a decline in traditional software methodologies in game development, there are companies that still use the waterfall method (Kristiadi et al., 2019).

Moving onto cross-disciplinary interaction, the results shown in Table 7 showcase that the participants’ disciplines usually have contact with Design and Programming, and rarely interact with Audio or Writing. From the answers obtained from the survey, it can be said that communication across different disciplines, depends on the workflow of the person. Getting in touch with different disciplines usually occurs when developers’ work aligns and depends on other disciplines. One of the participants who works in design responded that their collaboration with other disciplines is associated with their work:

I mainly have contact with programming since we are collaborating on the mechanics and systems of the game and art since I am involved in level design and visual guidance aspects of the game.

However, how the disciplines interact with each other shows a connection between the work location type and the studio size. Scheduling video meetings, and communication through text chats is common with remote work, while sharing the same floor with other disciplines allows developers to walk to each other’s desks and talk.

From the data obtained from the survey, having daily stand-ups is the most commonly used development strategy. All of the participants mentioned that daily stand-ups allow everyone in the development team to discuss how everyone’s progress is going and if there are any blockers or questions to be discussed before starting work. In bigger teams, there might be cases where the disciplines are split up because doing a daily stand-up every morning with a team of a hundred developers can take quite a long time to go through. Splitting up into separate teams that perform daily stand-ups, reduces the time spent on discussing and allows the developers to go back to work more quickly. Performing weekly stand-ups with the whole team once in a while to catch up with other disciplines or dedicating a day where every feature team comes together to sync up (Fane, 2018) can help in discussing problems, and dependencies.
Although performing daily stand-ups is not the only thing that contributes to cross-disciplinary communication. Having managers and producers who are knowledgeable in how each discipline works, can help in forming better communication between different disciplines. Being open in terms of communication with outside sources (Hesser, 2017), especially if the studio has investors and producers from outside of the company, can help producers bring in their aid towards the development team. In addition to the managerial improvements, doing “show and tell” sessions during sprint reviews, where the developers go over what they have done throughout a sprint, what kind of challenges they have encountered and how they have solved them. This can inform developers of what kind of issues can come up in different disciplines and allow the team to discuss how they can prevent them in the upcoming sprints.

Even though agile methodologies focus on minimizing documentation, there is always a use case where having some documentation is useful. Apart from syncing and managerial changes, one of the suggestions from the survey for improving cross-disciplinary communication was creating better documentation to ensure a coherent experience for the players:

Would be nice to have other teams create better documentation on what they work on so that we create something coherent for the players. For example, it would be nice to have a wiki on characters so that no one deviates from their personality.

In the Agile Manifesto, one of the four values is (Beck, et al., 2001) “working software over comprehensive documentation”. This value can remain true even when studios use some documentation to help developers with some of their workflow.

While the results of the survey show that %80 of the participants’ studios modify development methodologies. In video game studios, methodologies are seen more as guidelines, rather than strict rules to follow. As previously mentioned, the daily scrum was utilised differently depending on the studio size; some steps of Scrum need to be reworked to be used by bigger teams. The modifications can also vary with the game itself, for example, stages of the development can differ, because some game projects that act as a live service constantly need to be updated with new features, in-game events, and improvements to continue their life cycle. While other projects, that do not provide a live service, might have different stages to go through before the game’s release.

Even though studios make changes to development strategies, %80 of the participants in the survey concluded that it does not make a difference in cross-disciplinary communication. Since the interactions between developers from different disciplines mostly depend on the tasks at hand, their workflow does not change in a way that will require them to talk to other disciplines. The type of work location does not seem to affect this situation as well. One of the participants who works as an animator on-site, gave an example of why there is rarely an opportunity to work with one of the sub-disciplines in their studio:

Environment art. We rarely animated world placed objects so we don’t really need to collaborate that often.

Another example is given by one of the participants who works in the programming discipline:

As a gameplay focused Programmer I work closely with Design, Art, Backend, and QA. My current work place is mainly remote so communication takes place either in text chats or scheduled video meetings. Talks with Design is usually focused on how features should work and feel. Talks with Art is varied - often about syncing needs/requirements, getting deliveries, and iterating on visuals. Talks with Backend tend to be about defining requirements and designing APIs. Talks with QA will usually be about syncing what features need testing and reproduction of bugs.
Interaction with other disciplines that are not mentioned in the quote, like Audio, can come in later during development. Nevertheless, a remark can be made that certain professions require more collaborations to be made with different disciplines.

5.4 Overview of Patterns Found

For the last section of this chapter, the findings of the content analysis are presented with a focus on the patterns found between GDC talks, the field study and the survey.

In Table 8, the patterns obtained from the content analysis are shown with the source and the cells represent how impactful these patterns have been on cross-disciplinary communication. The impact was measured by how often the patterns were mentioned in the sources and the analysis of the statements regarding cross-disciplinary communication.

**Table 8** Sources and the patterns found with content analysis; cells marked with (+++) show a strong impact, (+++) show a regular impact, (+) show a minimal impact, (-) show that the pattern is not mentioned in the source.

<table>
<thead>
<tr>
<th>Sources / Patterns Affecting Cross-Disciplinary Communication</th>
<th>GDC</th>
<th>Field Study</th>
<th>Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of the video game studio</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Overlapping of the work</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Meetings that give insight into what the developers will do</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Having regular sessions for syncing</td>
<td>+++</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Asking for help</td>
<td>++</td>
<td>+++</td>
<td>-</td>
</tr>
<tr>
<td>Strategies that help when disagreements occur</td>
<td>+++</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Experimenting with different methods for planning sprints</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
</tr>
</tbody>
</table>
Based on the results shown in Table 8, it can be derived with some certainty that cross-disciplinary communication is affected by:

- The size of the video game studio
- The overlap of tasks, developers from different disciplines working on one core feature
- Meetings where developers from different disciplines participate
- Experimenting with different methods to find a structure that works best for the team’s needs

Three of the listed patterns can be connected with modifications made to agile development strategies.

Occurrences of overlapping tasks with other disciplines, allow developers to adapt to different workflows (Fane, 2018), and collaborate on features. For small teams, communicating with developers from different disciplines can be trivial such as walking from one workstation to another. However, in larger-scale studios, splitting the whole of the development team into small feature teams (Fane, 2018; Keith, 2020) can be the solution.

Conducting meetings that allow disciplines to check in with one another, whether through weekly stand-ups, or the scheduled big sprint days (Fane, 2018).

Overall, experimenting with different applications of development strategies throughout sprints can give insight into which implementation would suit the development team the best. There are no strict rules in agile development, such as every discipline in game development should do method A and then method B; the flexibility of this software methodology, allows development teams to come up with their methods on what works well in their case.

From the obtained patterns, a generalisation can be made that modifications of agile development strategies are likely to occur depending on the studio size.
6 Conclusions

6.1 Summary
This study aimed to explore how agile development methodologies can have an impact on cross-disciplinary communication. The research was conducted in three parts; first, GDC talks related to “agile development”, “project frameworks”, and “team communication” were analysed to give insights on how different video game studios tackle issues, and resolve conflicts with what kind of development strategies. Second, a field study was conducted with the usage of participatory and direct observation. Lastly, a survey was designed and shared with game developers to obtain more data focusing specifically on the relationship between communication across disciplines and agile development strategies.

A content analysis of each part of the research was performed to find patterns that are mentioned in all of the parts.

The results of the analysis showed that cross-disciplinary communication is affected by meetings where developers from different disciplines participate, the number of tasks that overlap with different disciplines and experimentations made on different methodologies.

Overall, the study contributes to research related to applied game development with some knowledge about how adaptations made to agile methodologies impact cross-disciplinary communication in video game studios.

6.2 Discussion
This study aimed to contribute to the academic research regarding game development, by providing an analysis of the practical side of video games, with the GDC talks, field study and survey. Since the research on game development in academics is limited, the findings in this research may highlight that there is potential to conduct studies in a practical area that can provide insight into how the video game industry functions.

6.2.1 Validity of Previous Research
It can also be said that the findings from the previous related research complement the findings from this research as well.

The switch from traditional software methodologies, like waterfall, to iterative methodologies such as agile development, is still ongoing and the results of the field study and the survey show that the usage of traditional methodologies is decreasing (Koutonen and Leppänen, 2013; Kristiadi et al., 2019).

The issues mentioned in the findings of Petrillo, et al. (2008) such as managing scopes, scheduling, crunch, and budget, are still being mentioned in recent development cases as well (Keith, 2020; Whitson, 2020). However, switching to an iterative way of developing games, and finding the methodologies that work best for the team to reduce the amount of encountered issues (Keith, 2020).

6.2.2 Ethical and Social Aspects
For the field study, a data collection agreement was signed between me and the studio, that allowed me to collect data mentioned throughout the results. This agreement clarified that the duration of the fieldwork would only be two months, and no other data would be collected apart from cross-disciplinary communication and agile methodologies.

For the survey, it was designed and prepared in a way that any personal information regarding the participants was kept anonymous. The data that will be collected from the survey was
mentioned in the survey to make it clear which information will be written and mentioned throughout the thesis. The participants were allowed to drop out of the study at any point in time and if they wanted to get their data deleted, they were allowed to contact me through my educational email.

### 6.2.3 Limitations

Certain limitations should be considered when interpreting the findings of the research.

The findings of the research have limited generalizability. Studios differ in size, culture, disciplines, and applied strategies. The focus point of the research was narrow and only the data obtained from a few studios were presented. As seen in the video by Paolo from TOKYO (2023), in one of the biggest video game studios, Capcom (2023), the developers roam around the studio to talk about certain parts of the game. Even though it may be more common for smaller studios to walk up to their colleagues' workstations, some big companies may do the same as well.

It should be noted that the games industry is secretive. Many of the strategies utilised in video game studios are not open to the public (Engström, 2020). Game developer postmortems and talks given at conferences are highly structured and biased. This impacts the GDC talks section of the thesis, the topics mentioned by the presenters are chosen strategically and only a specific set of examples regarding the topic are mentioned throughout the talk.

The field study showed a lot of connections between cross-disciplinary communication and applied agile methodologies, however, certain remarks must be discussed. First, all the field notes were taken by me, and me only, since there was no other researcher present in the studio this would suggest that there might have been instances that occurred that may have been missed out on. Second, since the time of the study covered four sprints, the workload and the time spent taking the notes may have varied throughout the two months. There might be some days, where more field notes were taken, and some days less.

Lastly, for the survey, the usage of snowball sampling worked well within the given period of time for the thesis, however, there are a few points that need to be mentioned with it. The survey was shared within game development communities that I was a part of, the audience reached was limited to the shared communities. This may also suggest that there is a potential for a sample bias throughout the study.

### 6.3 Future Work

This study can be further improved by analysing game developer postmortems, conducting in-depth interviews with game developers and reaching out to developers who were not a part of the study, such as solo developers.

Similar to analysing GDC talks, gathering data from game developer postmortems, would help in obtaining more data regarding the topic. The postmortems give out reflections of game developers in the industry which allows researchers to get insights about their experiences. However, researchers would need to analyse the data gathered from the postmortems, since industry sources are not considered academic:

> Industry sources cannot be used directly as academic results. However, they serve as important sources of information on the game industry’s perspective. Researchers can aggregate and analyse data from such sources.

Engström (2020, p. 152)

Conducting in-depth interviews about agile development methodologies and cross-disciplinary communication would provide more detailed discussions and reflections on the
topics. As a researcher, having interviews would lead to natural discussions with developers, and obtain more concrete answers by asking “how and why” regularly.

Broadening the audience by conducting research with different-sized companies would contribute to obtaining a more generalised result. In addition to various-sized studios, gathering data from solo developers about frameworks and communication would highly likely yield interesting results since one developer would take on the role of many disciplines. The development process would show many differences between teams developing games and only one developer developing games.
References

Atari, Inc. (2023) About Atari, Available at: https://atari.com/pages/about [Accessed 20 February 2023]


Atlassian. (2023d) What is scrum and how to get started. Available at: https://www.atlassian.com/agile/scrum [Accessed 27 January 2023]

Bay J. (2023). Which game job is right for you? 30 interviews with pro game devs, Available at: https://www.gameindustrycareer.com/quest-for-your-video-game-career/ [Accessed 5 August 2023]


Dataspelsbranschen (2022) Game developer index 2022. Available at: https://dataspelsbranschen.se/game-developer-index [Accessed 5 February 2023]

Dataspelsbranschen (2023) Game developer index 2023. Available at: https://dataspelsbranschen.se/game-developer-index [Accessed 12 October 2023]


Finji. (2023) Finji. Available at: https://finji.co/ [Accessed 3 August 2023]


Harmonix (2017) *SingSpace* [Virtual Reality Game]. Harmonix.


Hess, J. (2017). Improving the culture of critique: Communicating across disciplines, [video online], Available at: https://youtu.be/Mvz8MqQfzUk [Accessed 22 January 2023]


Informa PLC. (2023) *GDC vault.* Available at: https://www.gdcvault.com/ [Accessed 26 February 2023]

Informa PLC. (2023) *GDC.* Available at: https://gdconf.com/ [Accessed 26 February 2023]

Interactive Software Federation of Europe. (2022) *The Industry - ISFE.* Available at: https://www.isfe.eu/game-industry/ [Accessed 20 February 2023]


Keith, C. (2020) Taming the chaos: Lessons in project management, [video online], Available at: https://youtu.be/M0uuDsjy4bo [Accessed 24 February 2023]


Paolo fromTOKYO. (2023) Day in the Life of a Japanese Street Fighter 6 Game Planner. [video online] Available at: https://youtu.be/M6hf4Ll_WkE?si=qqvJfEUrHSiPySFW [Accessed 6 August 2023]


Square Enix. (2023) Final Fantasy [Computer Game Franchise]. Square Enix.


Appendix A - Survey

Thesis Project: Development Methodologies and Their Effect on Communication

Hello, my name is Selin Kaya, I'm currently a master's student in University of Skövde, doing my thesis work about investigating how agile development adaptations affect the communication between team members from different disciplines in video game studios.

This survey is for my master's thesis and is designed to help me understand which software development methodologies are used and modified to improve the communication between different disciplines in the video game industry. The results of this survey will provide an overview of which methodologies are frequently used and help me make connections in which parts of the agile development planning is modified, and the reasoning behind these changes.

This survey is expected to take about 10 - 15 minutes to complete.

By answering the survey, you agree that the answers you have given to the questions listed in this form can be used in my thesis. The information shared in the survey is the only personal data connected to you as an individual in my thesis. The information collected will be mentioned in my thesis and will be published in DiVA Portal (Digitala Vetenskapliga Arkivet). The one who will have access to the survey data is me. With the purpose of examination and guidance; exceptions can be made for the supervisors and examiners in the course: Master Degree Project in Informatics with a Specialization in Game Development A2E.

The only personal questions in this survey is related to:

- The video game studio size you work at
- Your discipline in the studio you work at
- The work location type you have (on-site, hybrid, remote)
- Software methodologies used in the studio you work at

All information will be saved until October 30, 2023. All data collected will be used anonymously.

Participation to this survey is completely voluntary and can be cancelled at any time. If the session is cancelled, all the data connected to the survey will not be saved.

If you have questions regarding this survey, you may contact me at c21selka@student.his.se.
Section 1: Basic Information

1. What is the size of the studio that you are working at? Choose the closest to the given ranges.
   - Solo Developer
   - ≤ 10
   - ≤ 30
   - ≤ 50
   - ≤ 100
   - ≤ 250
   - ≤ 500
   - ≤ 1,000
   - ≤ 5,000
   - ≤ 10,000
   - > 10,000

2. What is your work location type?
   - On-site
   - Hybrid
   - Remote
   - “Other”

3. Which discipline (e.g., Art, Audio, Design, Programming, etc.) in the studio are you a part of?

Section 2: Development Methodologies

4. Which development strategies are used in the studio?
   - Agile
   - Scrum
   - Waterfall
   - Kanban
   - DevOps
   - “Other”

5. Which development strategies does your discipline use?
   - Agile
   - Scrum
   - Waterfall
   - Kanban
   - DevOps
   - “Other”

Section 3: Communications Across Disciplines

6. Which disciplines in the studio do you regularly contact with, how and why? For example, "My discipline usually works closely with designers. We share the same floor as them, so it is easy to go up to my colleague and talk about the feature we're implementing."

7. Which disciplines in the studio do you rarely contact with, how and why?
For example, "In some cases, we look over a feature with programmers. We usually schedule a meeting with an email."

8. Do you think that the software methodology being used right now helps the communication in any way? In what way is it helping, or in what way it is not?
   For example, "Having daily standups are very helpful on seeing which teams are working on which feature."

9. Do you have any suggestions that can improve the communications across disciplines?

10. Does your studio modify the development methodologies in any way?
    For example: "Mainly scrum development methodologies are used, however, the sprint durations are 6 weeks, rather than 2 weeks." , "Since the studio consists of a lot of employees, we always have the sprint retrospectives online on a video call." , "We define definition of done as: If it's in the stable build, working as intended, then it's complete. Rather than seeing if the cards/tasks are done."
    o Yes
    o No

11. In what way are the modifications made?

12. Do you think the methodology adaptations make a difference in communication with other disciplines in the studio?
    For example, "After shifting from having sprints for 2 weeks to 4 weeks, we saw that it helped the tool programmers to add a new feature that the animators can use the next sprint."
    o Yes
    o No

13. What differences occur in terms of communication?

Why does it not make a difference? What would you expect to change with the adaptations?