Bachelor Thesis Medieninformatik

Rapid Game Prototyping

An Agile Approach to Early Game Development

---

submitted by

Mia Rose Winter

Student ID 41527

at Hochschule der Medien Stuttgart

on 04.08.2025

to obtain the academic degree of Bachelor of Science

First Supervisor Prof. Don-Oliver Matthies
Second Supervisor Carsten Kiess

Ehrenwörtliche Erklärung

Hiermit versichere ich, Mia Rose Winter, ehrenwörtlich, dass ich die vorliegende Bachelorarbeit mit dem Titel:

„Rapid Game Prototyping: An Agile Approach to Early Game Development“

selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Ebenso sind alle Stellen, die mit Hilfe eines KI-basierten Schreibwerkzeugs erstellt oder überarbeitet wurden, kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§ 24 Abs. 2 Bachelor-SPO) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.

Stuttgart, den 04.08.2025

Mia Rose Winter

Abstract

Game development is still a relatively young domain of software development, which faces many challenges. Cancellations, cost-overruns, delays and commercial failures are common experiences. Many issues may be attributed to failures in developing core game concepts and validating assumptions, leading to costly adjustments late in development delaying projects and affecting release quality. As preproduction is pivotal, informing decisions even late in production, it has the highest potential for risk-mitigation and warrants inspecting practices inspired by other industries and how they may affect game projects. As game development is software development, agile is used as a starting off point, being modified to facilitate rapid prototype to put increased emphasis on preproduction and maturing the game concept early. The approach was applied to the student project “Chainlinked” and its impact on product quality and developer sentiment was observed. The flat hierarchies that are enabled by this contributed to team members considering their impact on the product more meaningful and the thorough testing of the game concept did deliver a high quality product. However, insight into how rapid game prototyping affected the project was limited, due to the setting and missing prior experiences by team members, which would have allowed comparison in key ways. Regardless the approach is promising, and rapid prototyping has been proven in other domains, so further refining of the practice and applying it to more projects is justified.

Kurzfassung

Gamesentwicklung ist immer noch ein junges feld in der Softwareentwicklung mit vielen Herausforderungen. Viele Projecte werden abgebrochen, leiden Kostenüberschreitungen or verfehlen kommerzielle Ziele. Viele dieser Probleme können als Fehlschläge das fundamentale Spielerebnis oder Annahmen zu validieren, was zu kostspieligen anpassungen in der späteren Produktion führt, wodurch Deadlines verpasst und die Produktqualität beinträchtigt wird. Die Vorproduktionsphase is essentiell, denn Entscheidungen früh in der Entwicklung haben Einflüsse bis spät in die Projektlaufbahn und haben das größte potential Risiken zu minimieren, weswegen ein blick auf Entwicklungspraktiken anderer Produktbranchen gerechtfertig ist, um zu sehen wie diese Gamesprojekte beinflussen könnten. Da Gamesentwicklung ein Feld der Softarentwicklung ist, kann die Agile Entwicklung als Anhaltspunkt verwendet werden, um dann mit Rapidem Prototypisieren den Fokus auf die Vorproduktion zu erhöhen, welches ein frühes ausreifen des Spielekonzeptes ermöglicht. Dieser Ansatz wurde im Studentenprojekt „Chainlinked“ angewendet und sein Einfluss auf Produktqualität und dem generellen Empfinden der Entwickler untersucht. Die flachen Hierarchien die dadurch eingesetzt wurden haben dazu geführt, das Teammitglieder ihren Einfluss auf das Produkt als Bedeutungsvoll wahrgenommen haben, und das gründliche testen hat ein hochqualitatives Game produziert. Jedoch Einblicke in wiefern Rapides Game Prototypisieren das Projekt beeinflusst hat wurde eingeschränkt durch die Umstände des Projekts und der limitierten Vorerfahrung der Projektmiglieder, welche es nicht möglich machte den Ansatz mit vorherigen in diversen Kernkompetenzen zu vergleichen. Trotzdem ist der Entwicklungsansatz vielversprechend, da sich Rapides Prototypisieren in anderen Disziplinen bewährt hat, dementsprechend ist das Anwenden und Verbessern dieser Entwicklungsmethode auf mehr Projekte gerechtfertigt.

Contents

Acronyms

***DoD* – Definition of Done**

***DLC* – Downloadable Content**: Traditionally additional content for a game that may be purchased separately. Originally as games were distributed on physical medium, later additions to those was only possible by either distributing updates on additional physical distributions or, with increase in internet availability, by making those additions downloadable through various means and platforms. This lead to the now commonly used term of downloadable content, and is currently the primary way to extend a games content with physical media falling out of fashion.

***GDD* – Game Design Document**

***MVP* – Minimum Viable Product**

***NPC* – Non-Player Character**: An entity within a game, which is controlled by algorithms rather than a player, who players may interact with.

***PO* – Product Owner**

***QA* – Quality Assurance**: The practice of ensuring requirements are met and features are well tested. May also refer to a department tasked with executing quality assurance.

***SC* – Scrum Coach**: Traditionally referred to as Scrum Master. Changed to Scrum Coach for more inclusive language.

***UI* – User Interface**

Glossary

bug: A software or design issue that needs to be fixed.

developer: A participant in the team working on tasks. This definition is not limited to software developers, but extends to artists and other experts.

feature creep: The practice of ever-escalating project scope that is considered decremental to a projects success, as it may cause delays and increase complexity, as well as changing the original product vision in unintended, unchecked, or unapproved ways.

game engine: A sofware framework which often includes a suite of tools that generalize common technolgies used to develop game software products. Those most commonly include facilities to launch software on a variety of platforms, render UI elements, as well as 2D and 3D assets. Engines may be build with wide applicability in mind, most commonly Unity, Godot and Unreal Engine do target a variety of games. May also be specialized to certain genres or visual styles like GameMaker which exclusively focuses on 2D games.

stakeholder: A party with investment in a project. This can be leadership in a company or an external customer that comissioned the project, but can also extend to development teams themselves.

Figures

1. Introduction

Game development posses many unique challenges, as it is not only software development, but touches many other domains as well. While the industry has grown massively during the last decades, and teams have scaled to hundreds of participants, more and more project cancellations and disastrous game launches have be seen over the years. Poor management, cost overruns, missed deadlines and game ideas that are simply not engaging or interesting to its target audience are frequently named as causes. Modern development and management approaches are needed that answer these challenges and a good first step is looking at the early phases of game projects, the preproduction. Decisions and assumptions early in a projects lifetime can be pivotal for its success and the earlier issues are discovered, the easier it is to address them, before significant resources are wasted.

Initially, the game development process – and consequently current game development practices – and challenges need to be identified. Next, looking at the solutions found in wider software development and understanding the reasons behind them will provide a solid foundation for any approach going forward. Here Agile and more specifically Scrum will be examined. As those alone are not perfectly suited for game development without adjustment, other domains are analyzed, including lean startup and rapid customer feedback, as well as rapid prototyping. After understanding the challenges they seek to address, all of these approaches contribute to the approach of rapid game prototyping, aiming to optimize early game development in order to address the challenges identified with game development.

In order to evaluate the viability of the approach, rapid game prototyping is applied to a project with industry-like conditions. The game project “Chainlinked” gathers 19 university students in a self-organized lecture aimed at developing a game in a real world inspired manner, which has previously operated on traditional development practices enabling comparison between past development practices and this one. The impact that rapid game prototyping has on that project will be examined and the final product is being presented to publicly for playtesting, and evaluated in its quality. Additionally, sentiment by the development team itself is analyzed, to see if developers feel an impact of the development approach on the game as a result, as well as on their work individually and as a team.

Finally these results are put forwards judging rapid game prototyping on its impact, and how other projects may apply it in the future, what challenges it did and did not solve and how to potentially address them in future iterations.

2. Traditional Game Development

2.1. Software as Multimedia, Interdisciplinary Product

In Software Engineering regularly the products being developed go farther than code organization and architecture. Frequently many different domains and domain knowledge is needed in order to deliver a satisfactory user experience, intuitive and accessible design, and to implement all features and experiences asked for by stakeholders. Video game software and game projects therefore belong to one of the most complicated and multifaceted software products, often containing all the aforementioned requirements, as well as domains like sound and music design, world design, cinematography, story telling, game theory, game design, and potentially many more[1]. Because of this, most game project teams are made up of experts from many different disciplines, many of who are not well versed in software architecture and development. This presents multiple challenges for effective project management and planing, communication between these domain experts is vital and projects may fail not only because of flawed design, poor quality or misunderstood requirements, but also simply because they are not entertaining to players.

On top of that, many systems in game projects need to integrate with each other. For example a story teller may envision a NPC, which is to be modeled by a 3D artist, textured by another one, rigged and animated by a third, put in engine by a software developer and tested by a game tester. These long production pipelines are difficult to manage, touching many disciplines, where technical limitations or other factors late in the pipeline need to be known early in the chain of production to make informed decisions. Because of this, estimating trajectories and delivery timelines frequently becomes very hard to manage as well.

[1]Christopher M. Kanode and Hisham M. Haddad, "Software Engineering Challenges in Game Development," in *2009 Sixth International Conference on Information Technology: New Generations*; , vol. 0; , 2009; , 260–265.

2.2. Division of Labor through Specialized Departments

As game projects oftentimes have large teams with many different members proficient in different fields, those are commonly grouped together into specialized departments, each consisting of domain knowledge in certain fields. Graphic designers or 3D artists may be grouped in a graphics department, software developers into an engineering department, and so on. This way the product pipelines become streamlined, a game design department develops a Game Design Document (GDD) for a feature, the graphics department creates visual assets and models for it, then hands it off to the engineering department implement it into the game, to finally be tested by Quality Assurance (QA). Commonly, the entire project is overseen by a project management department consisting of key individuals managing the logistics and creative direction of the game, including leaders of each department and potentially even stakeholders.

This approach is easily scalable, an important requirement for game projects, as they can increase and decrease in members over the Projects Lifetime.

2.3. Lifecycle of a Game Project

As games and game development teams are varied, only three development phases can be generalized that are present in most projects. Depending on the scope of the project, the size of the team, or the project's timeline and requirements, more phases may be identified, but the following are the most important.

2.3.1. Game Design Preproduction

At the beginning of most game product pipelines as well as the Project in it's entirety, stands a preproduction phase that conceptualizes the basic game premise, identifies gameplay loops, lays out project scope and resources needed. Regularly prototypes are developed in this phase to try out various assumptions about the target market and how envisioned systems can work together and against each other.

During this phase the game project team can be very small, as the focus is focused on game design. Sometimes, experts from the other commonly needed fields down the line might not be present at all, especially if prototypes are very simplified visually and on a technical level – i.e. paper-made prototypes for a card game system. This allows for this early phase to be comparatively low cost, as fewer members are needed and thus

the smaller team is potentially more agile in decision making with a flat hierarchy and fast decision making.

2.3.2. Asset and Code Production

The second phase of production might scale the project team significantly. Many of the previously envisioned systems will not only need to be implemented, they will require many different assets and even more in-depth design. Additionally, a lot of software will be developed, even if the project chooses pre-existing technology like an established game engine, those can require modification and many systems will be needed to be implemented on top of that. Depending on the complexities of the project, different departments might be more important or have more members. As the game is being developed, a lot of testing can also be done on the systems as they are being developed, to collect feedback and adjust them if needed. On more complex projects with many different interacting systems, only at this stage can comprehensive testing be conducted, which might alter a lot of the design plans as assumptions from the preproduction phase might be proven incorrect or incomplete.

As games themselves are media products which might require hours, to in some cases hundreds of hours of playtime, a lot of issues with systems that the player interacts with might only now be discovered in this development phase. Adjusting existing designs is also more costly during this step, since changes to the design might make many already produced assets unnecessary, effectively having wasted potentially hundreds of hours put into their development, implementation and validation.

2.3.3. QA and Release

In this project phase the focus is on testing, fixing software bugs and improving the quality of the product. Changes to design or systems are rare and only done if necessary, as they would be very expensive. In this phase production departments are downsized or assigned to different projects, as not much production is taking place anymore. The only exception is Quality Assurance (QA), although in the industry generally external companies are hired to assist testing and keep costs low. This creates additional communication overhead as the downsized team may need to work together with external contractors to fix issues in the product close to release.

It is also possible that the product is released during this phase, and post-release software patches are produced as customers discover issues. This is often needed regardless of how thorough the validation was, as potentially millions of player can discover very

obscure issues with the system, and the deployment – especially on personal computers – has to deal with an enormous variety of hardware configurations, that themselves can cause issues with the software.

If the project is particularly successful it's also not uncommon to extend the product with requested features or sets of content. Developing those may be easier as smaller teams can produce those additions. Such expansions or DLC are most commonly monetized to justify the additional development cost. In rare occasions companies also choose to make these additions free of charge, either because initial reception was poor due to major issues with the product on release or to invest in the brand of the product as intellectual property and increased potential of success of future installments. A recent example of this is the 10 year anniversary celebratory month launched by paradox interactive that saw many free additions to City Skylines. The games successor had already released a year prior, but it's quality had been criticized and required many post-launch patches[2]. With the comments made by Paradox members around this celebration about it's sequel, these free updates to the acclaimed first installment could be seen as such an investment to improve the brands reputation.

2.4. Challenges of Department-Approach

Dividing teams into specialized departments brings with it challenges that many don't – or only partially – overcome. First, there is a great need for communication, inside the departments, themselves as well as between them. Frequently the extra levels of hierarchy cause information to be lost when going through departments. Due to them being specialized and the people in them being from different domains, communicating requirements and understanding the limits and needs of the different fields can be difficult as well, making cross-department cooperation difficult.

The division of teams into departments can also cause the people in them to isolate themselves from the rest, losing the bigger picture of the product and being uncooperative with the others, especially when communication and handing off of work between departments is centralized in the leading roles inside departments, creating further separation between project team members. This effect is especially prevalent in game projects, where a feature can go through many departments, leading the later depart-

[2]Nic Reuben, "Paradox Celebrate 10 Years of Cities: Skylines with a Roadmap of Anniversary Goodies;" (Rock Paper Shotgun, March 10, 2025); , last modified March 10, 2025, accessed August 3, 2025, https://www.rockpapershotgun.com/paradox-celebrate-10-years-of-cities-skylines-with-a-roadmap-of-anniversary-goodies.

ments to be very disconnected from the early ones, making informed decisions on scope and design challenging.

Without a clear and broad communication structure and team unity, departments that are often late in a production pipeline, like quality assurance, can foster resentment if they are often given unrealistic deadlines or scopes. As such, larger teams require a lot of high level organization and planing, the project leadership often becomes its' own department, which then causes even more separation between leadership and individual developers due to the steeper hierarchy. As a result, project management can have more difficulty assessing the state of the project or creating realistic goals and deadlines and expectations for stakeholders.

These steep hierarchies also cause the effort investment of "lower" team members to be decreased, as they understand themselves to be only a small part in a big production without much agency or creative input.

2.5. Importance of Preproduction: Early Risk Mitigation

As changes to features, systems and assets become more costly the longer a project progresses, the more important preproduction becomes. The earlier design flaws are discovered and assumptions are verified, the more costs can be saved. The risk of late-stage discovery is especially large in game projects that have high costs and take many years in development, as cost overruns and missed deadlines become likely whenever issues are discovered or assumptions are proven wrong, and entire production pipelines have to be restarted to fix them.

Game development also poses the unique challenge that it is a relatively young field, with limited knowledge about the underlying psychological effects on customers, often changing best-practices, as well as subjective evaluations of entertainment as the market changes often and customer expectations shift, potentially requiring many adjustments to the project even if all goes to plan.

Additionally, as the teams size increases over the project lifetime there is an increased risk of requirements not fully being communicated to new members or even entire new departments, further worsening potential communication issues. Because of this, not only is a good starting concept for the game important, but also comprehensive and validated documentation.

2.5.1. Risk of Cost-Overruns and Project Cancellations

Game projects often miss deadlines and end up costing a lot more than initially projected[3]. In recent years many news articles of cancelled game projects and closed game studios surface on the regular[4]. Frequently the reason for these closures and cancellation is attributed to rapid changes in the market that made these projects not suitable to further persue. Commonly these come in waves, as failed releases prove certain assumptions many studios had wrong, making the projects build on these assumptions likely to fail themselves. As games are costly projects therefore it's often decided to minimize risk by cancelling them. Frequently validation through game testing also reveals that assumptions were wrong or the game concept not engaging or entertaining, leading to delays and cost-overruns as production is ramped up again to address these.

Some project cancellations have also been attributed to the development time being too long, leading to the underlying game premise or cost structure being outdated by the time of release. Notably, the high profile failure of the game *Concord* has been attributed to its' basic game premise being very popular in 2017 to 2019, but by the time of release unsuitable for the current market, with many competing products released years prior already achieving market saturation.[5] This has lead to a shift in the games' publisher Sony to reconsider many in-development projects, with many being cancelled as a consequence. This has lead to millions in development cost being lost, developers being layed of and even entire game studios being closed down permanently.

[3]Fabio Petrillo et al., "Houston, We Have a Problem...: A Survey of Actual Problems in Computer Games Development," in ; , 2008; , 707–711.

[4]Wikipedia, "2022–2025 Video Game Industry Layoffs;" , last modified 2025, accessed July 26, 2025, https://en.wikipedia.org/wiki/2022%E2%80%932025_video_game_industry_layoffs.

[5]Matt Kim, "Why Concord Failed;" (IGN Entertainment Inc., September 9, 2024); , last modified September 9, 2024, accessed July 26, 2025, https://www.ign.com/articles/concord-victim-long-development-time-playstation; Keza MacDonald, "Sony's Big-Budget Hero Shooter Concord Failed Spectacularly – Here's Where It Went Wrong;" (Guardian News & Media Limited, September 11, 2024); , last modified September 11, 2024, accessed July 26, 2025, https://www.theguardian.com/games/article/2024/sep/11/sonys-hero-shooter-concord-failed-spectacularly-heres-where-it-went-wrong; Chris Scullion, "Sony President, COO and CFO Hiroki Totoki Has Shared His Views on Why Concord Was a Failure.;" (1981 Media Ltd., November 8, 2024); , last modified November 8, 2024, accessed July 26, 2025, https://www.videogameschronicle.com/news/sony-president-reflects-on-why-concord-failed/.

2.5.2. Difficulties Adapting to User-Needs and lack of Feedback

Two competing development structures in games highlight certain upsides and downsides of early feedback in game projects. Traditional commercial products are developed and tested internally, only being revealed to the public shortly before release, with feedback from it only being incorporated in future installments. In contrast to this the recent trend of the early-access games highlights how early and continuous customer feedback can improve the quality of the product and make it more resilient to market changes. In contrast to the previous long running project *concord* that released to a market that had moved on from its core premise, long developed games like *Satisfactory* or *Hades* have become huge successes – especially on official release – as continuos adaption with the market and customers during their development lifetimes aided in releasing the best products possible.

This is not an indication that early-access is inherently improving games, many more projects have shown the downsides of the approach. Game concepts that are not already well received on initial publication often cause sustained disinterest in the product and may require significant changes to the games' premise. Because of many abandoned game projects, the customer expectation for early-access games has become higher over time, so it can only support already well received game premises, but can't help a project that was ill-adjusted from the beginning. The emphasis here is again that a good and thorough preproduction and validation phase is vital to any game project.

3. Agile (Software) Development

In 2001 a group of software developers came together to define the Agile Manifesto and the twelve principles of Agile Software[6]. The group consisted of representatives from Extreme Programming, SCRUM, Feature-Driven Development "and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes"[7].

3.1. Motivation

As Game Development can be broadly understood as software development, examining common industry best practice in that domain should be examined. The challenges faced by software development are familiar to game projects as well and through understanding how those are addressed, a development approach for games can be build.

Agile is answering the need for software projects that need to adjust to changing requirements or a changing market. Agile seeks to constantly validate the state of the project to market conditions and stakeholder expectations. Through an iterative approach with regular examination, validation and aligning expectations agile development seeks to improve the quality of products empirically with well defined events, roles and quality control structures.

3.2. Core Principles

The Agile Manifesto defines multiple core principles that centralize the importance of delivering working software frequently, early and regularly.[8] It aims to empower people and teams, promote individuality and creativity. It emphasizes the need for direct communication, including between developers and business people.

[6]Kent Beck et al., "Manifesto for Agile Software Development;" , last modified 2001, accessed July 16, 2025, https://agilemanifesto.org/.

[7]Jim Highsmith, "History: The Agile Manifesto;" , par. 1; , last modified 2001, accessed July 16, 2025, https://agilemanifesto.org/history.html.

[8]Kent Beck et al., "Principles behind the Agile Manifesto;" , last modified 2001, accessed July 16, 2025, https://agilemanifesto.org/principles.html.

3.3. SCRUM

Scrum is a simple and powerful framework that provides a guideline for management while leaving the actual development to the teams themselves[9]. The team is trusted to know it's abilities best and as it is responsible for implementing product requirements, it is empowered to self-manage doing so.

The Agile development framework Scrum is formalized in the Scrum Guide[10]. The Scrum Guide contains the definition of Scrum and describes a set of events and roles that help teams develop products.

3.3.1. Roles, Events and Products

In Scrum there are three roles. Trust is put into any of these roles to work to the best of their abilities to reach the project goal. Because of this, classic project management is not required.

These are the Scrum roles summed up[11]:

- **Scrum Coach (SC)**: Oversees the Scrum processes. Coaches other Team members and makes sure Scrum rules are followed.
- **Product Owner (PO)**: In charge of the product goal. Creates requirements and determines the next Product Features.
- **Developer Team**: Takes on an appropriate set of product features and implements them under their own management. Estimates requirements and supports the PO when questions about the product come up.

At the core of Scrum lays an iterative approach to development, that identifies two circles, the **24h-Inspection** and the **Iteration**[12]. These events are identified by the Scrum Guide as the Daily Scrum and the Sprint. In the [Daily], according to Schwaber, the team synchronizes on what work lies ahead of them, and then is left alone to work on their tasks. At the heart of Scrum is the Iteration; each of them the team sets out to implement requirements, determine how to implement functionality, and modify their approach "as [they] encounter[] new complexities, difficulties, and surprises".

[9]Roman Pichler and Stefan Roock, eds., *Agile Entwicklungspraktiken Mit Scrum*; , 1st ed.; (Heidelberg, Germany: dpunkt, 2011).

[10]Ken Schwaber and Jeff Sutherland, "The Scrum Guide;" , last modified 2020, accessed July 10, 2025, https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf.

[11]Saso Nikolov, *Scrum Master Exam Vorbereitung*; (North Charleston, SC: Createspace Independent Publishing Platform, 2017); , p. 7-8.

[12]Ken Schwaber, *Agile Project Management with Scrum*; (Redmond, WA: Microsoft Press, 2004); , p. 5-6.

The requirements are maintained in the **Product Backlog** by the PO[13]. The PO is responsible for maintaining the tasks in the Backlog and for prioritizing them. At the start of the project the Backlog is filled with initial assumptions and evolves over the project as new requirements are discovered or the product vision changes, the PO dynamically identifying what the product needs.

The **Sprint Backlog** belongs to the Scrum Team and defines the work required for implementing a Product Backlog Item that has been selected for that Sprint as a Sprint Goal[14]. Initially during Sprint Planing the Backlog is filled with enough Tasks for the Team to work on for the next few days, each task should be no longer than a day in scope. The Sprint Backlog is owned by the team and may only be changed by them, it serves to make all work during the Sprint transparent.

At the end of the Sprint the Review and Retrospective take place[15]. In the **Sprint Review** all work performed is presented to stakeholders and progress towards the product goal is discussed. This event can also serve to adjust the Product Backlog, adding new items or refining existing ones as requirements and potential implementation become more clear. The **Sprint Retrospective** is for the team to reflect on the sprint and formalize improvements to their processes. Problems encountered during the sprint should be discussed and solutions shared and documented. If improvements to the product can be identified those are added to the Product Backlog as well. The whole process can be seen summarized in Figure 1. Notably, adjustments to the Product Backlog may happen during any part of the process whenever requirements become clear or change.

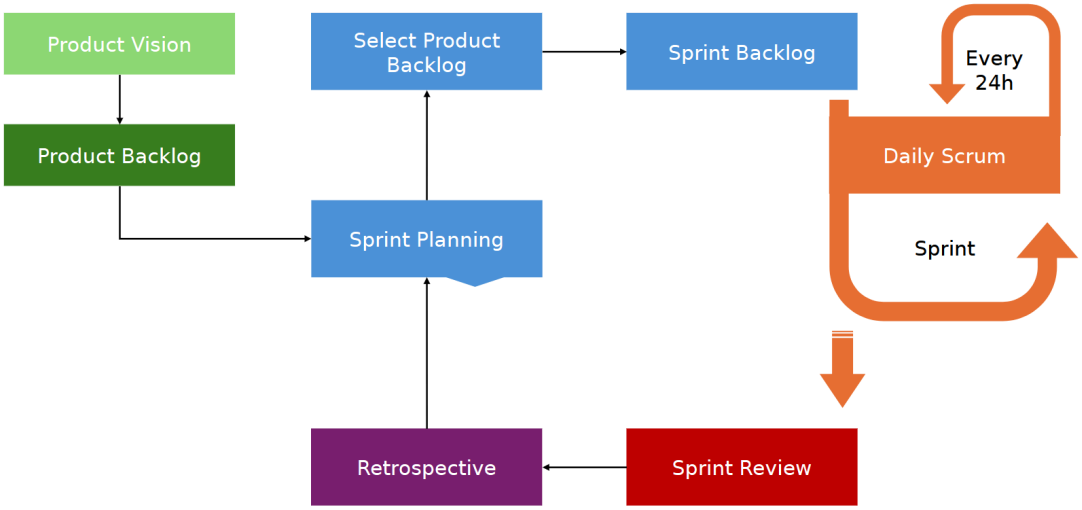


Figure 1: Scrum Flow Diagram

[13]ibid., p. 10.
[14]ibid., p. 11.
[15]Schwaber and Sutherland, "The Scrum Guide," p. 9.

Lastly, for quality control a **Definition of Done (DoD)** is created for product increments and tasks performed[16]. The DoD can contain quality requirements like passed tests or documentation, but also other requirements like reasonably fast response time.

3.3.2. Potential Benefits for Game Development

Scrum and Agile provide mechanisms for transparency and promote regular validation. At the end of each sprint, decisions about the future of the product will be made which allows projects to pivot to changes in the market or update features based on assumptions that have been proven or disproven. The small team size of Scrum promotes transparency between the team members and reduces overhead, as the different domain experts are encouraged to talk to each other directly, cutting down production time and reducing misunderstandings and overlooked requirements. These can be vital benefits to a game project which needs to handle a wide array of media domains and requirements. It also encourages creativity as everyone in the team is a valuable member whose contributions are seen, and who are given a chance to meaningfully take part in shaping the product.

A common claim about Scrum is the promotion of creativity. While not easy to prove there is evidence that the autonomous nature of the members of a Scrum team does improve and promote it[17] – in that case it would greatly benefit game development. While game products are software, they are also creative expression and entertainment. In the work of Conboy, Wang, and Fitzgerald we can see an important criticism in that "The concept of an on-site developer who travels to the customer's area of work to get a true feeling of what the customer wants and does on a day-to-day basis has received no attention"[18], which is something game projects cannot overlook and have already implemented a diversity of mitigations for. As game projects are entertainment products the concept of live demonstrations, conventions and early access has opened up opportunities to early engagement with the product. Especially, events like conventions can be very valuable as members of the project team can experience how prospective customers experience their product, seeing first hand what difficulties they face and determine what improvements they can make. While not part of the default Agile framework, heavy consideration should be put on this as it could be of great benefit to a game-focused Agile approach.

[16]Nikolov, *Scrum Master Exam Vorbereitung*, p. 8.

[17]Kieran Conboy, Xiaofeng Wang, and Brian Fitzgerald, "Creativity in Agile Systems Development: A Literature Review," in *Information Systems–Creativity and Innovation in Small and Medium-Sized Enterprises: IFIP WG 8.2 International Conference, Creativesme 2009, Guimarães, Portugal, June 21-24, 2009. Proceedings*; , 2009; , p. 124f..

[18]ibid., p. 128.

3.3.3. Limitations

As game projects touch many different domains, finding a DoD will be difficult. A DoD for code-centric projects would contain code quality standards, code reviews, passed tests and regular refactors, while a 2D sprite goes through several stages of concept, to line art, to coloration, to testing in-engine which will inevitably have other processes and mechanics on how to determine "done". Because of this, different DoDs might be necessary for different domains, or collecting common assumptions into a general DoD while domain-specific requirements get their own sub-sections. Regardless, a team in a game project will have many focuses, with experts from different industries that are used to different work and validation workflows, so the DoD early on will need a lot of collaboration and iteration. This is built into Scrum as part of the retrospective workflow, but it is a consideration that the mechanics for ensuring quality will be limited early in the project until the DoD matured enough.

As the complexity of the game product rises, so does the length and complexity of asset pipelines. A 2D-Game will consist mostly of 2D sprite assets and moderately complex code depending on the technology used, like a pre-existing game engine. Once 3D is introduced, and/or image fidelity goes from a simple style to for example one focused on realism, the workload increases significantly as there are more domains and domain experts involved, more time is needed to develop individual parts, and more time needs to be put towards integrating them together in the game and validating them. This will be a challenge for the small-team approach of Scrum, and multiple Scrum teams may need to cover different parts of the game product, leading to larger scale and more advanced strategies like Scrum-of-Scrums which bundles multiple scrum teams together with a top level scrum team made up of representatives from each scrum team.[19]

[19], "Scrum of Scrums;" , accessed July 28, 2025, https://agilealliance.org/glossary/scrum-of-scrums/.

4. Rapid Game Prototyping

4.1. Motivation

When addressing the difficulties in game development, Agile alone cannot answer all of them. As game projects are not perfectly suited for a framework like scrum, which as one of its core principles prioritizes creating product increments and continuos releases of improved versions of the game, some goals of the approach and methods need to be modified while still keeping to the core motivation of Agile – transparency, empiricism and adaptability. We identified the project's preproduction phase as vital to its success, so we will look at other domains that provide answers on how to optimize this.

As our goal is early risk-mitigation targeting the preproduction phase, a particular technique is of interest, which addresses similar issues in other fields, rapid prototyping. While different domains have slightly different naming, shared methods and goals can be identified. Their similarities and how they differ lay the foundation for how to apply this to game development specifically.

4.2. Rapid Prototyping in other Domains

4.2.1. Traditional Manufacturing

Many manufacturing industries embrace rapid prototyping. It embraces shorter lead times by leveraging computer aided design with various technologies to produce prototypes quick and easy, instead of having to rely their iteration on external manufacturing[20].

Early on rapid prototyping has been embraced by industries like the auto industry, as it has long and in many cases multiple supply chains for various parts of their products, so more effective research and development of individual parts without having to involve suppliers immediately was greatly beneficial in improving quality and reducing cost and development time. The early manufacturing of prototype parts using Computer Aided Design (CAD) and cheap production technologies like 3D printing allows validating assumptions and aids communication within the development teams as well as between experts from different fields with each other[21]. A prototype can be presented to stake-

[20]Steve Upcraft and Richard Fletcher, "The Rapid Prototyping Technologies," *Assembly Automation*; 23, no. 4; (2003); : p. 318; , https://doi.org/10.1108/01445150310698634.

[21]ibid., p. 319.

holders early to validate specification, it can also be submitted for testing frequently to see if design decisions work out in reality as they have been envisioned. Parts can be tested for compatibility and functionality with other – potentially also prototyped – parts. Producing these prototypes using such technology is much faster and cheaper, and while the final product will be made using different, more reliable technology with higher quality materials, this early look at the part and the early testing helps minimize wasted time and money during the development process, by eliminating flaws that could not have been predicted iteratively as they come up.

While game projects rarely produce physical products – ignoring physical distribution media – they are made up of many complicated systems having to be integrated with each other. Additionally, they are an entertainment product and while theory and psychology can be applied to determine the engagement of systems, the combination of them needs validation early to minimize waste. These requirements are similar to the long supply chains of the automaker industry and the rapid validation with stakeholders and consumers can also be achieved here as well.

4.2.2. Lean Startup - MVP and Rapid Customer Feedback

At the core of the Lean Startup movement lays the desire to find the "product-market fit"[22]. It seeks to optimize the way new products are developed, to understand the needs of potential customers, understand them, and then develop solutions that fit their needs at great value.

This is achieved using the Lean Product Process, an iterative approach to understand a customer section, identify unmet needs, attempt to solve these needs and then test these solutions via prototypes with the customer, repeating and refining as development progresses[23]. Central here is the Minimum Viable Product (MVP), the minimum set of functionality that combined can be considered viable to the customer. When initially building a product **only** the MVP is being developed and tested, as it is believed that focusing on and developing it is ultimately what will let the product succeed and only afterwards are more features and more unmet needs can be taken into consideration, typically using the same iterative methodology. Important for this is the MVP prototype, testing out features and solutions before investing into their development which allows teams to learn and adapt faster.

[22]Dan Olsen, *The Lean Product Playbook*; (Nashville, TN: John Wiley & Sons, 2015); , p. xvii f.
[23]ibid., p. 8ff.

According to The Lean Product Playbook, every MVP prototype is a hypothesis[24]. A prototype can be the new product, improvements to an existing feature, or the creation of a new feature for an existing product, depending on what is tested different types of prototypes may be employed. Importantly, the prototype should capture the user experience as thoroughly as possible, "minimum" should not be taken as "just functional" but as a representative slice of the potential final product that is also reliable, usable and enjoyable to use. Lean identifies different types of MVP tests not just for testing functionality and completeness, but also for marketing and more.

Lean defines a variety of ways to test those prototypes[25]; quantitative and personal tests face-to-face may be employed, and oftentimes many of them should be employed simultaneously. The Lean Product Playbook gives a variety of examples on how to test out products, what questions to ask, and how to ask them. Any research and development process may find suitable testing processes in this guide regardless of industry or domain.

Using lean practices, new ideas and adaptions are tested early and repeatedly, with real potential customers in realistic environments with minimal developer interference to collect the most valuable data and feedback, guiding further improvements and adaptions and potentially new prototypes. Only once an MVP has been tested and validated to work with the prospective customer, to meet their needs and deliver a strong value-proposition is it being developed, as it is seen that the risk of the development process then has been minimized and the highest percentage of the target market will be interested in the product thus maximizing profit.

Many lessons can be drawn here for game development, as it can be seen as developing a product that needs a specific need of a target audience. That need might be entertainment generally, but should be more precise as it should target a specific customer audience, fans of a genre, a new market, or existing intellectual property. When identifying what type of game should be developed as a product, needs and expectations in that game need to be identified, solutions developed and then tested via prototyping if the solutions work as a product. This way the game iteratively and empirically improves over time to maximize its value-proposition to potential customers, which is most likely includes game cohesion, entertainment and a fair price which requires optimal production cost which will be aided by this process.

Lean Startup shows how capturing needs and ideas, testing those ideas with quick and targeted prototypes, and building on the findings from those tests, can help develop

[24]ibid., p. 89f.
[25]ibid., p. 143-165.

a product fast and precise that will need it's target markets needs. The suitability of this process to Agile and Scrum is highlighted by Olsen as Chapter 12 of his book[26] goes into detail on how to facilitate Agile methodologies and scrum to develop the product once the MVP has been proven – furthermore reinforcing the compatibility and comparability of these approaches.

4.3. Applying Rapid Prototyping to Game Development

Seeing the benefits of early and frequent prototyping in these areas makes its potential for game projects apparent. Validating ideas and features early, minimizing cost and risk alongside with aligning expectations and requirements with stakeholders early are potential vital improvements to the development process.

As rapid prototyping is very agile in nature, with its iterations and empirical approach to product development, it lends itself well to the agile development framework Scrum with some adjustments.

4.3.1. SCRUM as a guide Framework for Prototyping Process

To use Scrum in the game development process to guide the rapid prototype framework a few adjustments must be made. Scrum as a framework is not a definite playbook on how to develop products so we can use the concepts and ideas it puts forwards and slot in ideas and products from rapid prototyping to adapt it to the rapid game prototyping approach.

4.3.2. Experience instead of Product Increments

A central product in scrum is the product increment. Scrum seeks to produce at least one every sprint and every product backlog item becomes a cornerstone to the product increment. Traditionally, such increments are understood to be product features or improvements, this can take the form of new processes, code produced or visuals enhanced. As the goal in preproduction in Rapid Game Prototyping is not to produce the actual product, but develop the ideas behind it, this is not the sort of increment that should be looked for here. The instruments for this are prototypes, which can be

[26] ibid., p. 201ff.

software or simply paper representations of game mechanics – as long as they are able to be discarded after every sprint if necessary – to look at new ideas or iterate on old ones without being slowed down by old code or assets.

Keeping that in mind, the increment looked for every sprint is not a physical product, but the idea behind the product. Because of this, one vital document in game development comes into focus, the Game Design Document (GDD). The GDD is a piece of documentation in game development that describes in-depth various mechanics or features of a game, various components and systems, the rules that it operates on, the way the player interacts with it, or the game in its entirety.

The target of a sprint becomes improving the idea of the game, envisioning new gameplay features and validating them – in isolation as well as together with the game as a whole. Consequently, the increment becomes improvements to the GDD and accordingly, product backlog items and the DoD should always target it. Increments become changes and additions to the GDD rather than any piece of code or assets produced, and the DoD focuses on when an idea and description in the GDD is tested, validated and fully documented.

4.3.3. Prototypes produce Increments

At the center of Rapid Prototyping stand the prototypes. They are not products produced but serve to test assumptions and design decisions, to communicate with others internally as well as stakeholders and prospective customers. As our increments are improvements to the GDD and agile provides empiricism, new ideas need to be analyzed individually when applied to the GDD. As systems can have complex interactions, changes to the GDD should be kept to a minimum with each new iteration to ensure any changes noticeable to the player experience and game system interactions can then be reliably attributed to that change, and can inform further changes and adjustments.

However, there has to also only be one change every sprint. Rather, one *prototype* should implement only *one change*, so if multiple ideas stand in conflict with each other, or are different answers to the same question or problem statement, they not only can, but should, be developed into prototypes in parallel so they can be compared. This allows finding the best answer to a problem or assumption made, as multiple prototypes can be analyzed for pro's and con's, and the best solution can be chosen and added to the GDD.

This all ensures that any change to the GDD, any increment, is an informed and tested decision that is always validated against the game concept as a whole.

As systems become more complex or the concept more multi-faceted, it might be required to test systems and changes to them individually. When this is required, the division should always consider systems that directly interact with each other, and leave them as a coherent unit when trying out alterations. This ensures that the changes are not reviewed in isolation from the part of the game it will need to interact with. Regardless, even disconnected systems need to be tested as a unit, so once individual systems have been refined, a joined prototype needs to be developed combining all the new parts of the GDDs into one, to ensure the quality of the product as a whole.

4.3.4. Small, Interdisciplinary Teams instead of Departments

To produce prototypes that validate all the game mechanics and that can produce GDDs that contain all the information needed to later produce the systems, the teams working on them need to cover all the domains later needed to develop the game, otherwise informed decisions that touch various domains cannot be made. As such, the individual scrum teams need to be made up of all the experts required to develop this type of game. Further, if multiple prototypes will be worked on at the same time, multiple scrum teams should all each have all the domains needed covered, so they can operate independently to take full advantage of the agility of scrum.

Besides the already known benefits of scrum – achieving better communication, transparency and encouraging creativity – which are vital when discovering new problems and issues and coming up with potential solutions. This also ensures that all domains are considered when changes are made, because different experts closely work together. Rarely does a decision only affect one domain in game development, oftentimes different areas are connected, so having all the experts with the required knowledge working closely together will allow them to discuss the effects potential changes have with more depth, and evaluating prototypes will produce deeper and higher quality insights.

This also encourages cooperation in the development process, as developing the prototype requires individuals working together to build the systems instead of departments delegating tasks and ordering products from each other. When artists, designer and developer work closely together to build features and test them out, it deepens their understanding of each others domains, and makes considerations and difficulties transparent to them as well. Finally, the nature of the prototype most often requires the

team to regularly play the game they work on together, which can further improve teamwork.

Ultimately, while it may be departments or huge scrum structures that develop those systems during the production phase, with the individuals from the initial preproduction most likely still present in key positions or leadership roles, this can still benefit the cohesion and communication between those departments.

4.3.5. The Sprint Goal as Game Design Experiment

As the prototypes become instruments to inform increments of the GDD, testing individual changes and features, each sprint becomes an experiment. Each sprint the team or teams should each take an idea, or come up with an idea, on how to improve the overall game concept. This becomes a creative endeavor where either discovered problems will find solutions that are then tried out, or new features get envisioned that add to the existing systems. While control still lays with the PO this process can encourage every member of any team to contribute ideas, furthering team cohesion and expanding the pool of potential features. Developing multiple prototypes allows the opportunity to give opposing ideas a try that each address the same issue in different way, to then pick the best one out of them. The sprint goal may become a “what if”, or a “wouldn’t it be cool if” and the sprint seeks to provide an informed answer to that. This may naturally engage the team and help make the product more entertaining.

4.3.6. Maturing the Game Design Document throughout Sprints

Each sprint a new game idea, multiple ideas, or solutions to problems are proposed. Those need to be applied to the same base game every sprint so a central GDD is vital, that is always updated at the end of a sprint with the changes that have been proven to work best. When multiple changes are trialed they need to always be applied to the same version of the GDD to ensure the findings are consistent and comparable. This way over the course of preproduction and multiple sprints, the GDD will mature with more well validated features, issues encountered are fixed, and the game experience will become more entertaining.

Over time the comprehensiveness of the document itself needs to be validated as well, it needs to be handed off to new teams, external teams, or teams that have not worked on prototypes for a few sprints to test it out. If any issues are encountered with the document and questions are raised which are not answered by the GDD then those

need to be addressed and added to it. With these occasional validations and refinements the GDD is matured until external members can fully comprehend and implement the game's design without direct involvement of the teams creating it. This will be vital during production as team sizes and processes are scaled up and direct communication will become more difficult, as many different people will work in isolation on the same set of features, while still needing to be synchronized on mechanics, systems, intentions and the interactions between them.

5. Related Work

5.1. Game Scrum

The paper by Godoy and Ellen looks at many different Agile development approaches that have been adapted to game development[27]. It proposes an iterative approach to game development using Scrum, using the process thorough the entire project to improve the development process, keep the scope of the project reasonable and to optimize for the "fun-factor" of the game.

The paper looks at a project developing an educational game for learning software development, where game Scrum has been applied. In it's conclusion multiple problems are identified that were not solved by game Scrum, but the approach was judged to be promising and could be matured further. Notably, the paper discusses a prototype that had been developed in the preproduction phase of the game project that informed significant improvements to the game concept, but no note is taken on validating that concept further with more prototyping.

As the post-production section discusses that the game concept has been described as tedious it is reasonable to assume the game concept had more need for testing and validation.

5.2. Rapid Software Prototyping

The work by Luqi and Steigerwald seeks to improve the design process of software, inspired by rapid prototyping from other domains, it proposes computer driven rapid prototyping and looks at various technologies and languages that make prototyping software more effective[28].

The paper also discusses using an iterative approach to development, that is inherited from the rapid prototyping approach but may also serve as an early agile development approach, given that it was originally published in 1990, years before formalization of frameworks like Scrum. It claims that prototyping and rapid prototyping are in need

[27]André Godoy and Ellen F. Barbosa, "Game-Scrum: An Approach to Agile Game Development," in *Computing Track - Short Papers*; , SBC - Proceedings of SBGames 2010; (PO. Box 668, Zip Code 13560-970 – Sao Carlos (SP), Brazil: Instituto de Ciencias Matematicas e de Computacao (ICMC-USP), 2010).

[28]R. Steigerwald and Luqi, *Rapid Software Prototyping*; , Dudley Knox Library; (Monterey, CA 93943: Naval Postgraduate School, 1990); , https://hdl.handle.net/10945/43606.

of software aided programming, that the prototype stage is vital in developing systems and needs more focus and technology.

As the software development domain is related to game development, looking at the value of rapid prototyping in game projects is worthwhile.

- Agile, having its roots in Software development originally, seeks to make software products and projects adapt to ever changing markets and demands from shareholders, as well as shorten time to market
- Under that mold, the idea of using languages and technologies aimed at developing software prototypes fast have been envisioned
- Low code, no code, and generic UI frameworks help get design and processes done fast to make validation quicker and more direct

5.3. Agile Game Development

C. Keith remarks that the game industry has, through its fast growth over the last decade, neglected to build sustainable practices and is suffering loss of talent due to crunch and failed projects[29]. As game development projects have gone from individuals making games to large sized teams of specialists spending years developing projects a crisis in game development emerged, where "the pressure to follow ill-conceived plans leads to poor working environments".

The major difficulties Keith identifies are feature creep, over-optimistic schedules, as well as the unique challenges faced by the preproduction phase of game development[30]. The preproduction phase is concerned with finding the "fun of the game" that will guide the production phase. Production can begin when uncertainty about the game concept has been minimized, then teams can be scaled, production becomes predictable, and content can be produced. Ideally, preproduction is concluded at some point and production can start; in reality, the transition between those phases is more sliding as preproduction can be fairly unpredictable and decisions have to be made to start subsets of production in order to meet deadlines. However, when production is started too early there is a high risk of time being wasted by assets being produced with unclear requirements, which may then need to be changed or discarded later.

This further supports the optimizing preproduction in order to minimize risk is an important concern.

[29]Clinton Keith, *Agile Game Development*; , 2nd ed.; , Addison-Wesley Signature Series (Cohn); (Boston, MA: Addison Wesley, 2020); , p. 3f.

[30]ibid., p. 20ff.

6. Development of Project “Chainlinked”

To apply Rapid Game Prototyping to a game project the lecture “Game Praktikum” will be chosen. Traditionally, this lecture lets teams of 15 to 30 students from different fields develop a game using a classical, department based management approach. Because of this, changing the organization over to agile and rapid prototyping will yield the most potential insights into strengths and weaknesses of the approach.

6.1. Previous Development Approach

The lecture “Game Praktikum” sees many students of various study programs come together to self organize and develop a game. The project time is limited to one semester with a final presentation and public exhibit. Generally, this lecture is attended by 15 to 30 students and is facilitated by dividing the student body into multiple departments for the various disciplines, such as Engineering, Graphics, Game Design, QA, Publishing and Project Management. Each department elects one department leader, usually a student of a higher semester or a masters degree student and those leaders coordinate the project trajectory together with the project management department in weekly meetings, as well as weekly meetings of each department individually. Fixed milestone events are chosen to hit development targets like functioning prototype or feature completeness. The professor of the lecture acts as stakeholder and advisor, equivalent to a customer.

The classical project management structure has various controlling mechanisms to it, with regular weekly meetings and fixed milestone events, but it frequently faces various challenges, such as scope-creep, missed deadlines, and breakdowns of communication, all of which lead to various degradations of product quality. Generally the game concepts are also not validated enough and the product ends up with a mediocre potential to engage its players.

6.2. Applicability to General Game Development

As the project is self-organized with the professor in a stakeholder role, applicability to general game development is partially provided. The size of the teams is similar to small game development studios, and the makeup of students of various disciplines, as well as international students, mirrors possible configurations of such a team with experts from different backgrounds. However, the students in the lecture do not work on the project full time, it is one of many lectures they attend and thus can only invest so much time and commitment into it, unlike a game studio where each employee would likely be a full-time worker.

Because of the academic setting, some possible controlling mechanisms are also not present. In an industry setting leadership has more methods of sanctioning misaligned workers, as well as supporting them. There exists no equivalent in this setting to granting individual workers vacations or removing them from the project, as each student signed up for it has to stay with it because of university rules around practical projects.

Somewhat different as well are set deadlines. While delays in an industry setting are unfavorable and can cause monetary damage, it is generally possible to extend the development time of projects – this is not the case in this academic project setting, as examination deadlines are final and the only way to correct a project timeline is then to reduce the size of the scope. However, as Rapid Game Prototyping aims to improve the preproduction process and the definition of project scope, it potentially partially mitigates this issue.

Finally, the product is not monetized, which changes potential priorities about the marketability and structure of it. There are no investments made into paychecks or marketing, and no return of investment is required – so the project has potentially more creative freedom and can be more experimental than it would in an industry setting.

6.3. Project Initiation

The game development project is kicked off with two initial meetings. In the first the general guidelines for the project and lecture are presented to interested potential participants and everyone is put into small groups of three to five people each to brainstorm a potential game concept with a card based theming. In the second meeting all teams present their first game concepts and one of those is chosen through majority vote. After these initial events, participants enrolled in the project and the development process enters preproduction with the chosen game idea.

6.3.1. Scrum Team Setup

The final project team of 19 members is divided into three scrum teams with one PO and one agile coach. The three teams are made up as follows:

- Team 1 (white) has six members, the SC of the team is also a game design lead that assists the PO in creative decisions. The Team has a high number of artists.
- Team 2 (blue) has six members of mostly people experienced with software design and programming, which would advise on technical challenges.
- Team 3 (orange) has five members with various experiences that can take up any task.

The PO of the team has previous experience with the Game Praktikum and has industry experience with Scrum. The agile coach has lead various projects including the Game Praktikum multiple times, and has extensive game development experience and advises the entire team. Based on the schedules of team members two work days per week are decided, with one of those days shared for all teams to host Sprint Review, Retrospective and Planing. Sprints are set to be weekly encompassing those two work days per team. Besides the Scrum Dailys, all SCs, the PO and the agile coach also meet once weekly to align the general project trajectory if necessary, as well as to advice and teach the SCs.

6.3.2. Developing a co-operative cardgame as Product Goal

The core concept of the initially pitched game is identified and summarized in one sentence as the product goal. The most important attributes are the co-operative nature of the game, requiring players to work together to take down an enemy, as well as the focus on taking down one powerful enemy that itself is not controlled by a player and instead based on rules. An additional requirement is to keep play sessions short, making prototype playtesting faster, and allowing for the game to be played through in a reasonable amount of time when presented publicly, so players experience the entire game and get a full impression of it. A deck building mechanic is also proposed. The final product goal and game pitch is decided as “A co-op card game where 4 players collaborate to defeat a non-human boss and build their deck. A game should not last longer than 20 minutes”.

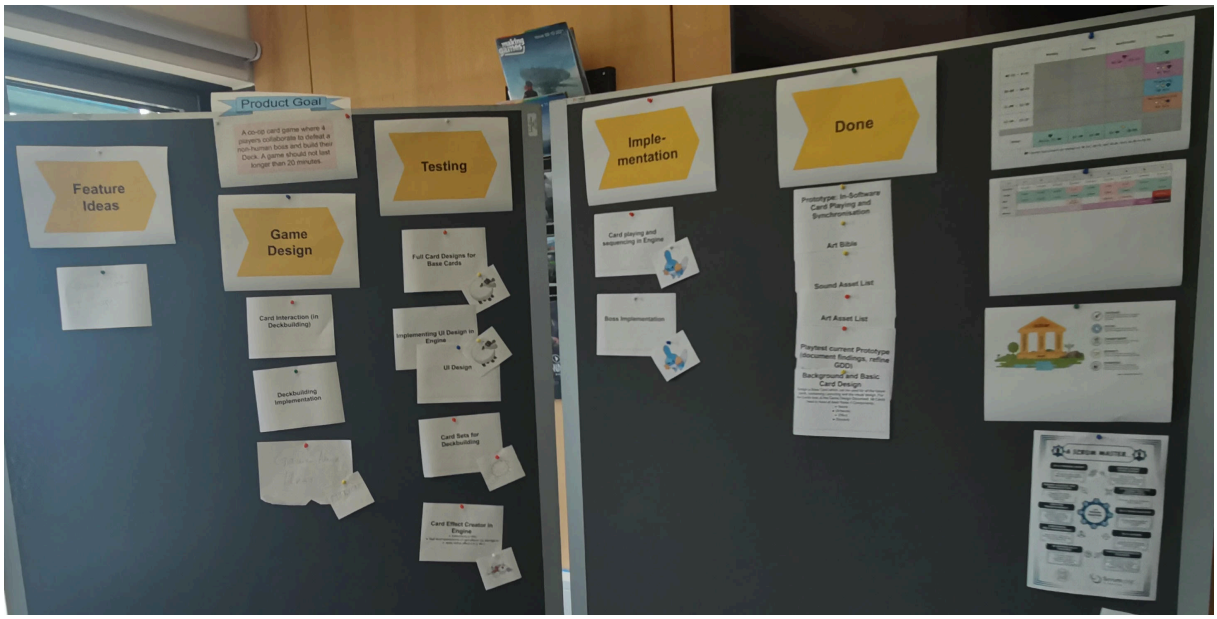


Figure 2: Shared agile wall used by the scrum teams (05/2025) with the product goal, product backlog and important timelines and schedules

The team as a whole is educated on scrum methodology and events by the agile coach, as well as rapid game prototyping.

6.3.3. Project Roadmap

As the game is set to be released at a fixed date, all other milestones have to be planned in relation to it. The total development time is limited to 16 week, between the first two weeks of project launch and the final public demonstration on the universities' open-campus event – called "MediaNight". With the project launch at the beginning of March, finalization of the GDD is targeted for April, with production slowly starting at the end of that month. This enables the entirety of May for production, which transitions into post-production in June. With the MediaNight event on the 3rd of July 2025 as the final deadline, a feature freeze is scheduled for two weeks before that day, to give enough time for finishing features and fixing bugs.

6.3.4. Leveraging Commonly Used And Familiar Tools and Platforms

With this restrictive timeline in mind, the team chooses to use tools many of them are either already familiar with, or are easy to learn. As the game engine Godot is decided, because it can produce working 2D games fast and its support for the C# programming language enables faster learning, as the language is either already familiar to most of the developers or easy to pick up, because many of them had learned Java in other lectures previously. Git is familiar to most developers as well and well supported by Godot, with a GitLab instance provided by the university as code and documentation repository. Artists are most familiar with the platform Miro which is used to create mood-boards and design guides, with final documentation to be added to the GitLab documentation repository. The PO is familiar with the graphic design software Figma and uses it to produce templates for the prototypes that can be printed and playtested. Lastly, the video and text chat platform Discord is chosen to allow for quick remote communication and remote meetings in case in-person appointments are not possible, while priority is still put on collaborative work in the projects offices space.

6.4. Preproduction

6.4.1. Using Weekly Sprints for Fast Prototype Iteration

To quickly iterate on game concepts and ideas, a sprint length of one week is chosen. As each team has two assigned working days per week this will give each one enough time to develop adjustments to the game design into a full physical paper prototype, playtest it, and collect feedback for further adjustments to the GDD. With three teams this allows for up to three prototypes being developed in parallel, allowing multiple different approaches for the same systems to be tested, to compare and pick the best one.

6.4.2. Prototype Evolutions and Results

With the basic game premise established, many different systems need to be developed. Focusing on resources available to the player, many different ideas on how players acquire and use cards need to be tested. Resource management needs to be optimized for balance, as well as to encourage and reward player co-operation; players need a reason for working together to not only justify the premise of a 4-Player game, but optimize engagement as well. The mechanics around the boss enemy are vital, taking into consideration the resources the boss employs, how its behavior is implemented and what to communicate to the players, as it is important to make the enemy comprehensible while not making its behavior too predictable. The system behind the desired deck building needs to be chosen, how players collect cards and build their deck, answering questions of if they share a deck – or does each player have their own, and are any cards freely chosen or do they fall into character classes that need to be decided beforehand. With the myriad of considerations and details of these systems, many prototyped will be developed and playtested.

First, basic resources for the players are decided, all players share the same pool of cards and each card falls into one of four elements .. with some cards creating and others consuming elements to trigger gameplay effects. After testing different turn-order mechanics the final system sees the players sharing a turn, with them each picking a card one after another to so implementing this in the game engine easier. Each player can choose a card and place it on one spot on a horizontal deck, left, right or in between cards by other players; encouraging co-operation and consideration for other players picks. Cards will have effects that either require enough elements to be present on turn to be most effective, as well as effects that are conditional or affect by cards to the left

or right of the card as it is placed on the board. This system is very complex to develop, but since it is the most fun of all the systems tested – and it gives the game a relatively unique game feel – it is considered to be worth the effort. This system also gives the game it's name, Chainlinked", as players chain together card effects. The boss enemy has it's own set of cards, without elements, to keep the system from being too overwhelming and to simplify its algorithms. After all four players have placed their cards, the boss adds a card of it's own onto the board, which will have various effects, from destroying other cards to attacking random players or players which meet a condition – like "player with lowest remaining health points". With this the total resources each player manages is their health bar, the enemies remaining health, the elements and cards in play, as well as the cards they can choose from. To further simplify and also speed up the game flow, each round the players get presented in turn with three cards at random from the players deck, and chose and place one of them. Other systems with players having cards in their possession, with and without trading, drawing and destroying them have been tried out, but are too complex to balance and implement in the short amount of development time.

6.4.3. Final Prototype and QA by Creating a Game Manual

With continued progress, less game ideas need to be tried out, enabling first production steps to be started. One of the teams is tasked with implementing a framework for the multiplayer system in the game engine, while the other two continue prototyping and start developing concept art and a design system.

Finally, the team that has spend a few weeks working on the multiplayer system is chosen to playtest the final prototype, without extra instruction, to find issues in the documentation and hidden assumptions that are not clarified. The goal is to develop the GDD into a full game manual like it would be in a released board game, that would solely be responsible for teaching prospective players on how to play the game, enabling later development efforts to be precise and requirements well understood. This step is vital and helps uncover many missing instructions, requirements, details, and mechanics in the GDD. It also serves as a final check to make sure all the systems in the game work well together and are engaging.

6.5. Production

6.5.1. Asset Pipeline

The white team that has most of the experienced artists and the lead game designer is tasked with creating a design guide, an art bible and menu wireframes and designs. Assets are first prototyped, then selected and finalized to be put into the game engine. Artists are tasked with adding the assets to the git repository, adding them in the Godot engine as resources – to make sure the in-engine representation is accurate. As assets become available they are marked in an asset list as completed, so other teams developing game systems can implement them.

6.5.2. Code Production and Validation

The blue and yellow team work on developing the game in the game engine. The blue team takes over the multiplayer code, and developing the basic card selection and gameplay mechanic. The yellow team focuses on menus, audio design and will support the white team in implementing the graphical assets, as well as animations. Each code change is being developed based on a common development state, each implemented feature is reviewed by at least one other developed and at the end of each sprint, the combined changes are tested and merged back into the development build. The development build is at all times functioning and playable according to the DoD, so that playtests can be performed on it to find bugs.

6.5.3. Documentation

While teams might work with platforms like Figma or Miro, or exchange information over discord, everything relevant to the projects production is required to be put into the documentation repository, as the projects singular source of truth. The GDD and all other descriptions of systems are documented there, all asset lists are linked there, and all assets that aren’t in the game repository themselves are uploaded there. This centralizes documentation and minimizes overhead by avoiding individual sources having to be searched for on various platforms. Additionally, this gives the teams the flexibility to choose their own workflow and tools, as they only need to put the results of their efforts into the documentation repository at the end.

6.5.4. Quality Assurance (QA)

With the production concluding, the final two weeks are fully dedicated to finishing up systems and polishing as a feature freeze is enforced. Not all systems prototyped were able to be implemented by the time the feature freeze arrived, with the deck building mechanic and others falling out of scope due to complexity. However, the basic system of each player in turn choosing cards, playing them in the execution chain, and the boss enemy placing their card, then the chain being executed, is fully implemented with a variety of cards and card effects. A lot of playtesting is taking place and many bugs are being fixed, putting the game in a polished state. Artists are also able to finish most assets for them to be put in engine. Sound design and animations are finalized optimizing the user experience further.

6.6. Release and Public Presentation

As the day of the open-campus event arrives, a game booth is set up with four computers each hosting one player in a shared session of the game. Alongside is a survey for players of the game to collect feedback for later assessment of the quality of the game, and the success of the rapid prototyping process. The final build is missing a few intended assets, but this was not noticed by most players as the overall presentation displayed a coherent design not being distracted by minor imperfections. Over the course of the evening many visitors tried out the game, either as entire groups or supported by team members to fill a session up to four players. The overall reception of the game is very positive, with many showing interest in the game.

6.7. Evaluation

6.7.1. Player Satisfaction

During the public demonstration eleven players filled out the provided user survey. Participants would choose for each question between five answers on a scale from “No” (1) to “Yes” (5). The average result of that survey can be seen in Figure 1.

Question	Mean
1. Did you like the game?	4,46
2. Did you have fun?	4.82
3. Do you feel you understood the UI?	3.55
4. Do you feel you understood the rules?	4.00
5. Do you feel you needed to work together with the other players?	4.36
6. Do you feel the game was difficult?	2.09
7. Do you feel the game took too long?	2.91
8. Do you feel the game had variety?	3.36
9. Would you play the game again?	4.27
10. Would you show or recommend the game to a friend?	4.36

Figure 1: Player Satisfaction Survey – Results

Overall, the reception of the game is very positive. Notably, the game was not considered difficult and the length of a game has been perceived as appropriate.

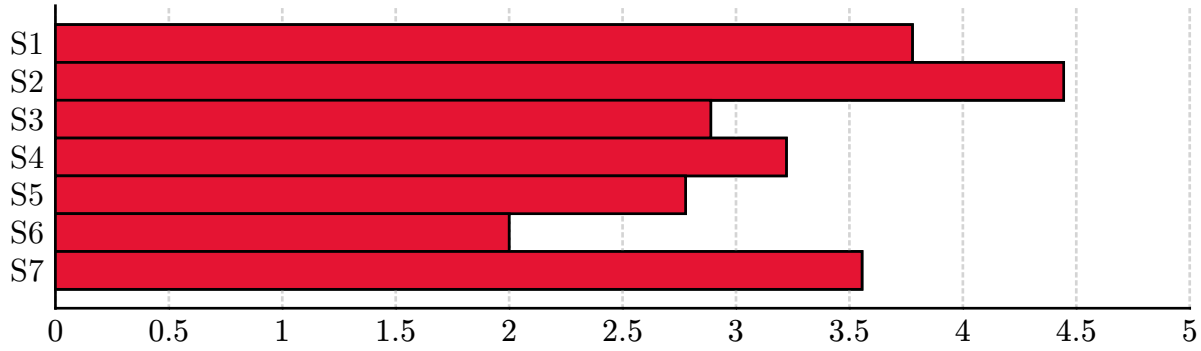
The perceived variety is close to 3.0, which indicates the game could potentially receive more systems or more content, considering that features like the boss rage mechanic were not implemented in time this potentially points to the prototype being more complete than the final game, which only implements a subset of it.

The User Interface (UI) is scored 3.55, indicating there is potential for better user experience – which may be a shortcoming of the preproduction – as the prototypes did not take the UI into consideration.

6.7.2. Team Satisfaction

The team survey had up to nine participants, later sets of statements were only shown if a participant selected if they had participated in specific team settings before and were hidden if not, as those statements asked a participant to compare the current project with previous experiences related to that. The number of participants answering the statements is denoted at the end of each title of set of statements. Each statement could be rated between “Disagree” (1) up to “Agree” (5), the graphs contain the average for each statement.

Please rate your experience with the project. Questions about other project members relate to your scrum team, rather than the entire team as a whole. [9 Participants]



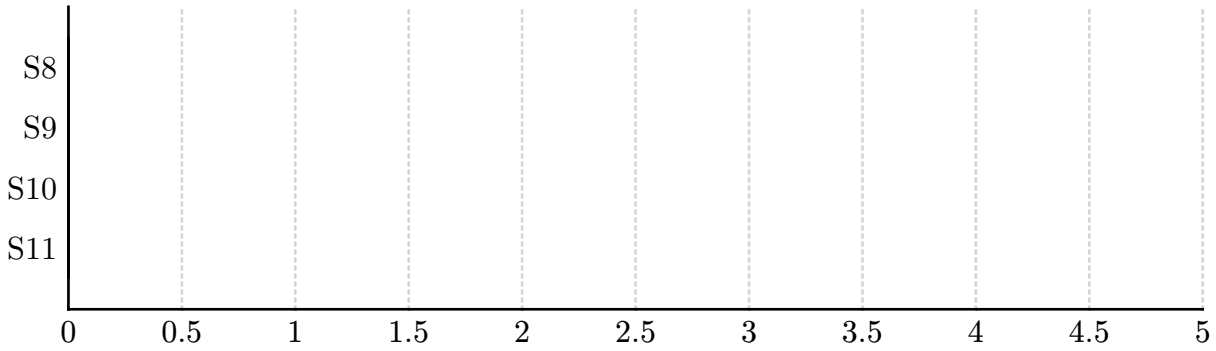
S1: I consider the Project successful
S2: I consider my contribution to the Project meaningful
S3: I consider everyone's contribution to the Project meaningful
S4: I always knew what I had to work on
S5: I always knew what my team members were working on
S6: I always knew the overall progress of the Project
S7: I am proud of the game we made

Figure 3: Team Satisfaction Survey - General

The first set of statements shown in Figure 3 aimed at assessing the quality of the game and the satisfaction with the final product. Most scores are above 2.5, indicating the project is considered mostly successful. Notably, most participants considered their contribution meaningful, which may be attributed to the small team sizes and flat hierarchies succeeding in enabling more contribution and influence on the game by individual team members.

On the other hand, the score for transparency in project progress is low around 2.0. This can either be because of the specific execution of the project, or because the rapid prototype approach makes it harder for the team to judge how close they are to completing the project scope, as it puts a larger emphasis on preproduction; lengthening it while cutting down on production time, obfuscating progress to the teams.

Please compare this Game Praktikum with previous once you have attended in relation to the Project Management Approach (previous department based approach vs. current scrum inspired approach) [0 Participants]

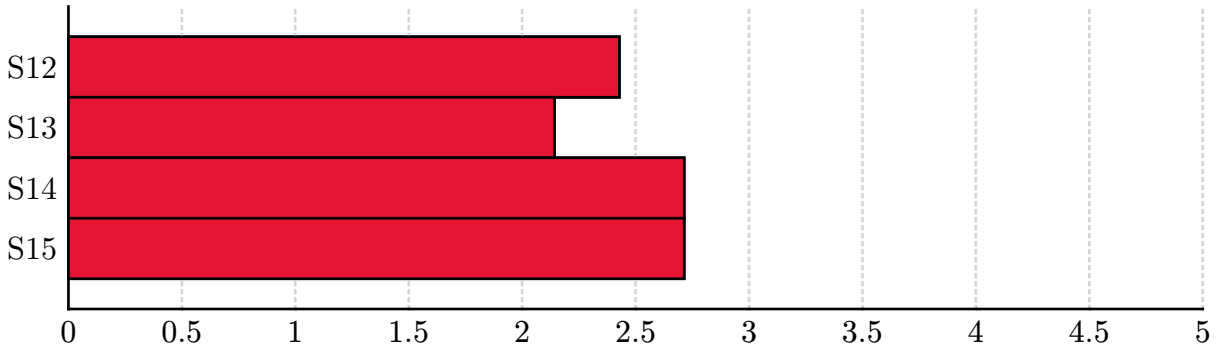


S8: I think the new method improved the quality of the Product
S9: I feel like the new approach improved my teams motivation
S10: I feel like the new approach improved my motivation
S11: I feel like the new approach made the project timeline more predictable

Figure 4: Team Satisfaction Survey - Game Development

Statements S8 to S11 in Figure 4 were aimed at members that have previously participated in the Game Praktikum lecture, but no participant was applicable for scoring these statements.

Please compare this Game Praktikum with previous Game Projects you have attended in relation to the Project Management Approach [7 Participants]



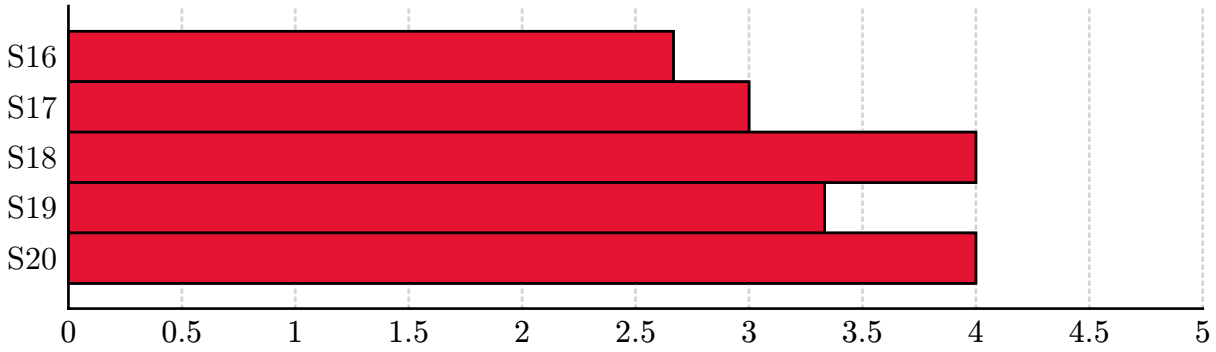
S12: I feel like the current approach made the game more fun
S13: I feel like the current approach made the game more polished
S14: I feel like the current approach made the game more understandable
S15: I feel like the current approach made the game better

Figure 5: Team Satisfaction Survey - Game Development

The statements in Figure 5 were shown to participants with prior game development experiences. All scores are around 2.5, which indicates that these team members did not perceive an increase in quality of the final product with the rapid game prototyping approach. They also did not see that the approach made the product worse, a slightly

lower score in the statement of game polishing might indicate that they perceived the two weeks post-production time to be too short, but the score is not low enough to be definite.

Please compare this Game Praktikum with previous larger (more than 5 people) team projects you attended in relation to the Project Management Approach [3 Participants]



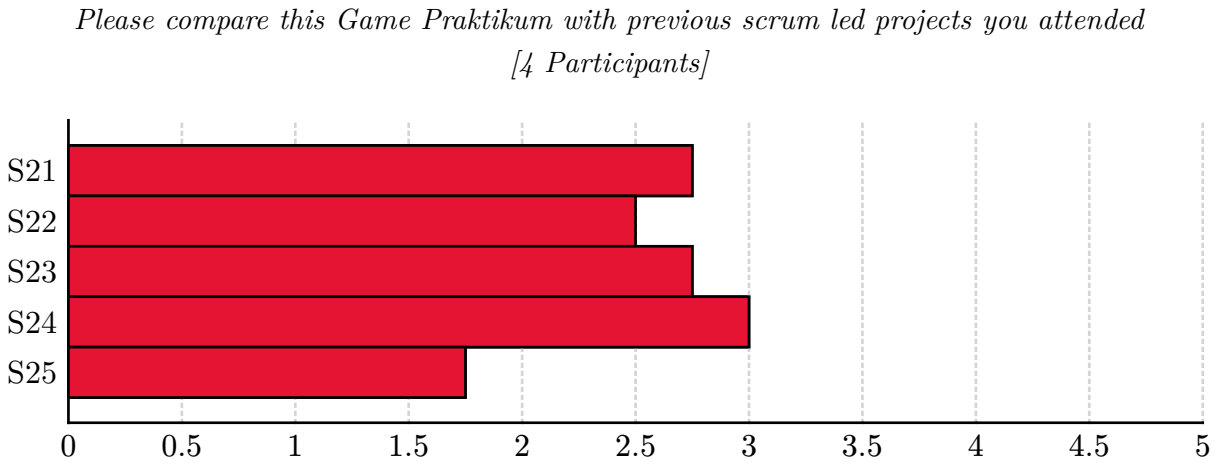
S16: I felt our teamwork was better than what I've seen previously
S17: I feel we always knew what everyone in our team was working on
S18: I felt the project management was always there without being overbearing
S19: I felt we had less, but better, meetings
S20: I felt there was little administrative overhead and I could focus on work more

Figure 6: Team Satisfaction Survey - Team Organization

The statements in Figure 6 were shown to participants with prior experience in larger teams. This is an important distinction to the previous statements of game projects, as larger teams significantly increase the challenges in game development. Low participation – with only three team members being able to score these statements – limits the applicability of these scores. Many lectures of the students require them to develop products in groups, but most lectures keep those group sizes below six members. The Game Praktikum is notable in team size not matched by other lectures, so this low participation is not notable.

The team members that have experience in larger teams, however, did perceive an improvement in project management presence and lower administrative overhead. This can be attributed to the flat hierarchy, as well as the nature of scrum generally succeeding. Each team having a SC and frequent communication with the PO, which both can be perceived as project management, is a potential explanation and would align with some of the values of scrum.

The lowest score is found in improved teamwork, but as it is still above 2.5 it is not reasonable to assume rapid game prototyping worsened the perception of this for these participants.



S21: It was easier to use scrum in this project
S22: Everyone in my team benefited from improved transparency provided by scrum
S23: The Sprint Reviews were effective in collecting feedback
S24: The (work-)daily standups improved teamwork
S25: My opinion of scrum improved

Figure 7: Team Satisfaction Survey - Scrum

The statements in Figure 7 were shown to participants with prior experience with scrum. Low participation here as well – with only four members being able to score them – limits applicability. Many students do not attend any lecture teaching or using scrum in projects, so most of these experiences are most likely from mandatory internships at software companies that most degrees require.

Most scores are around 2.5 indicating that participants did not perceive the execution of scrum in this project to be better or worse than what they have experienced prior. The last statement asking participants if their opinion of scrum improved is very low, below a score of 2.0 – or close to “disagree” – further reinforcing this conclusion.

6.7.3. Lessons Learned

While the final product is well received, high quality and well regarded by the teams and players alike, the development still encountered some notable challenges. Some project members perceived others to be working less than they were, which might either be because the scrum process wasn’t implemented perfectly – leading to work not being seeing by other team members – or the successful increase in transparency caused members to compare perceived workloads with others negatively affection perceived teamwork.

Many felt they were not able to assess the projects progress over its lifetime, which can be caused by the skewed production time towards more preproduction, as team members may be used to long production phases – this has given some the impression

the project was behind schedule most of the time. Efforts should be made to make the project progress as clear as possible, maybe using more milestones or estimating a completion like a game completion percentage. Judging these completion states might prove difficult, but even an inaccurate estimation might improve the perception of progress to the teams.

A challenge that had to be overcome early in the project was the aversion to Scrum by many developers, especially ones with prior experiences in software development. Potentially prior experiences with Scrum may had a degrading effect of Scrum as a concept in those team members – the project did not change this opinion. As this is a trend that can be generally observed in software development this is likely to be a common experience in any project implementing rapid game prototyping in the future. There were also project members with neutral or positive experiences with Scrum, but the negative sentiment was voiced more often affecting team morale. Insights from team members with prior Game Praktikum experience would have been valuable to judge if those feelings were justified, but are regrettably absent. Potential methods for implementing Scrum that increase enthusiasm for it is worthwhile to develop for future projects.

While having one team be more focused on art and design was perceived positively, having two teams tasked with development was times perceived has a hinderance. While the teams were well connected internally, having systems be developed by both teams that interacted with each other proved difficult, due to reduced communication and alignment between scrum teams. A significant challenge was the implementation of the peer-to-peer multiplayer framework – enabling the four-player gameplay to be played through four individual computers on a local network – as it was very complex and connected to almost all systems in the game. This complexity delayed progress and at some point even required handing over the task from one team to another, as progress was stalled – a shift in perspective was needed that ultimately saved the feature, but caused friction, delays and development overhead. Late in preproduction the alternative of having the game on a single computer, with support for four game controllers was pitched – which might have been the better choice – and may have been considered had it come up earlier. This illustrates that the rapid game prototyping approach with the paper prototypes lacked examination of technical requirements and assumption, software prototypes may have been necessary as well to address this, but were not feasible in the short development time and with the limited expertise by most team members in developing games.

6. Development of Project “Chainlinked”

A lot of scheduling issues arose during the project, partially because timetables of some students in teams not aligning well, making the planning of shared work days difficult, requiring short online daily scrums and remote work. Additionally, many of the days assigned to the lecture that saw every team present and enabled reviews and retrospectives was on Thursdays, a day that saw an unusually high number of holidays in the first half of 2025, causing a lot of missed workdays with limited participation and having to plan alternative review appointments that themselves saw very limited attendance, also because of scheduling conflicts with other lectures. Regrettably this is a unique issue of the academic setting and could not have been mitigated. Because of this rapid game prototyping may perform more efficiently in an actual industry setting with more work days available that are exclusively allocated to the project.

Additionally, the agile approach and regular playtesting saved a lot of time too. Most previous iterations of the Game Praktikum teams that had been formed into departments needed dedicated weekly meetings, as well as weekly meetings for the leaders of these departments, and initial team building to encourage communication between members of a department and the leaders of them. None of this was necessary this project as team members regularly engaged with each other by developing and testing the prototypes together, as well as daily scrums aligning the teams each work day instead of one long weekly meeting requiring the team to catch up on work performed and work planned. Usually the weekly leader meeting would also make decisions on the product, which was then communicated in the weekly department meetings, adding degrees of separation and regularly causing miscommunication in requirements, something avoided by the agile framework as product decisions were centralized on the PO and every team member was able to directly communicate with them.

7. Summary

After examining practices from software development and other domains, the combined approach of rapid game development was constructed that aimed at taking benefits from the agile development framework scrum, the idea of rapid prototyping and rapid iteration, as well as some lean practices and combine them to improve game development – especially preproduction. The preproduction phase of game projects was judged crucial to minimize risks and maximize product quality, consequently emphasis was put on making this project phase as effective as possible. A longer preproduction phase, organized using scrum methodology, aimed at validating the game concept and assumptions iteratively, until it was proven to be optimal. This was partially successful, because the longer preproduction phase did produce a thoroughly tested GDD, but the game was not implemented fully as there was not enough time during production. Because post-production was tightly scheduled, game polish may have been affected negatively as well, but the final product was of high quality regardless. When the game was playtested on the open-campus event most of its players considered it high quality and engaging, encouraging confidence in the rapid prototyping approach, as it relates to an improved game and gameplay experience. The development process itself faced many challenges, from scheduling conflicts due to different timetables of students and many work days aligning with holidays further limiting work performed. Sentiment towards scrum itself was also mixed before the project started and had not notably improved by the end, which should be a potential focus for improvement of future projects implementing rapid game prototyping. Regardless, most team members were proud of their contributions they made and the final product itself, indicating that overall the development experience and satisfaction has been improved by the approach.

8. Conclusion

Creating many prototypes and interrogating assumptions thoroughly, validating systems and features extensively, has produced a well regarded, high quality game. However, sentiment towards rapid game prototyping inside the project team have been mixed. Without insights into how this project compares to previous iterations, it is hard to assess if negative or ambivalent perceptions of the impact of rapid game prototyping on the game development process – and the quality of the product – are justified. Many team members did not have prior experience in larger teams or with scrum, limiting insights in how the development approach may have impacted the process as well. Ultimately, a high quality and engaging game was successfully produced, and the flat hierarchy of the approach likely had positive effects on team member engagement with the project and their perceived influence – potentially contributing to morale and product quality. Rapid game prototyping, which combines well known practices, has the potential to improve game development, as projects are matured early, proving marketability and enabling adjustment before significant resources and labor is invested. More projects should implement the approach, as well as agile development practices generally, to judge the influence of them more precisely. The quantity of research on game development practice is relatively limited and any additional insight can prove valuable. Future research should also look into how to improve developer acceptance of scrum and agile practices, as many seem to be negatively preconceive, limiting the benefits of the approach and causing friction. Efforts into making the overall progress of the project more transparent, if possible, would be valuable as well, as the novelty of the rapid prototyping approach shifts focus and time invested into preproduction, obscuring overall project completion to developers.

While comparison with past projects and their development practices was not possible through the experience of team members, the high quality of the final product can be considered indicative of rapid game prototyping having a positive impact. However, the project structure still differed in some areas from real-world game projects; some of the positive effects on development might be unique to the setting of the project. More implementations of rapid game prototyping, including outside of student projects, are needed to further identify its impact on development and product quality.

Appendix

Survey Data

User Survey

Fandest du du hast die Regeln verstanden?	Fandest du du musstest mit den anderen Spielern zusammenarbeiten?	Fandest du das Spiel ist schwer?	Fandest du das Spiel dauert lang?	Fandest du das Spiel ist abwechslungsreich?	Würdest du das Spiel nochmal Spielen wollen?	Würdest du das Spiel anderen zeigen/empfehlen?
1	4	1	2	2	5	5
5	5	4	2	4	4	4
4	3	2	1	4	4	5
5	5	2	4	2	5	5
4	5	2	3	3	4	4
3	5	2	3	4	5	5
5	5	3	3	4	5	5
5	5	2	2	4	5	5
4	5	2	4	5	5	5
5	5	2	3	3	4	4
3	1	1	5	2	1	1

Team Survey

Please rate your experience with the project. Questions about other project members relate to your scrum team, rather than the entire team as a whole.

I consider the Project successful	I consider my contribution to the Project meaningful	I consider everyone's contribution to the Project meaningful	I always knew what I had to work on	I always knew what my team members were working on	I always knew the overall progress of the Project	I am proud of the game we made
4	5	1	3	2	3	4
4	3	4	1	2	1	4
4	3	4	2	3	1	3
4	5	4	3	2	1	4
4	5	1	4	2	3	4
5	5	5	5	5	4	5
4	4	3	4	4	2	3
1	5	1	3	2	1	1
4	5	3	4	3	2	4

Please compare this Game Praktikum with previous once you have attended in relation to the Project Management Approach (previous department based approach vs. current scrum inspired approach)

I think the new method improved the quality of the Product	I feel like the new approach improved my teams motivation	I feel like the new approach improved my motivation	I feel like the new approach made the project timeline more predictable

Please compare this Game Praktikum with previous Game Projects you have attended in relation to the Project Management Approach

I feel like the current approach made the game more fun	I feel like the current approach made the game more polished	I feel like the current approach made the game more understandable	I feel like the current approach made the game better	I felt our teamwork was better than what I've seen previously
2	2	2	3	
3	2	3	4	
1	1	4	1	2
2	2	2	2	
				2
4	4	4	4	
1	1	1	1	
4	3	3	4	4

Please compare this Game Praktikum with previous larger (more than 5 people) team projects you attended in relation to the Project Management Approach

I feel we always knew what everyone in our team was working on	I felt the project management was always there without being overbearing	I felt we had less, but better, meetings	I felt there was little administrative overhead and I could focus on work more	It was easier to use scrum in this project
3	4	1	3	1
				5
2	5	4	4	3
				2
4	3	5	5	

Bibliography

. “Scrum of Scrums”. Accessed July 28, 2025. https://agilealliance.org/glossary/scrum-of-scrums/.

Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, et al. “Manifesto for Agile Software Development”. Last modified 2001. Accessed July 16, 2025. https://agilemanifesto.org/.

Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, et al. “Principles behind the Agile Manifesto”. Last modified 2001. Accessed July 16, 2025. https://agilemanifesto.org/principles.html.

Conboy, Kieran, Xiaofeng Wang, and Brian Fitzgerald. “Creativity in Agile Systems Development: A Literature Review.” In *Information Systems–Creativity and Innovation in Small and Medium-Sized Enterprises: IFIP WG 8.2 International Conference, Creativesme 2009, Guimarães, Portugal, June 21-24, 2009. Proceedings*, 122–134, 2009.

Godoy, André, and Ellen F. Barbosa. “Game-Scrum: An Approach to Agile Game Development.” In *Computing Track - Short Papers*. SBC - Proceedings of SBGames 2010. PO. Box 668, Zip Code 13560-970 – Sao Carlos (SP), Brazil: Instituto de Ciencias Matematicas e de Computacao (ICMC-USP), 2010.

Highsmith, Jim. “History: The Agile Manifesto”. Last modified 2001. Accessed July 16, 2025. https://agilemanifesto.org/history.html.

Kanode, Christopher M., and Hisham M. Haddad. “Software Engineering Challenges in Game Development.” In *2009 Sixth International Conference on Information Technology: New Generations*, 0:260–265, 2009.

Keith, Clinton. *Agile Game Development*. 2nd ed.. Addison-Wesley Signature Series (Cohn). Boston, MA: Addison Wesley, 2020.

Kim, Matt. “Why Concord Failed”. IGN Entertainment Inc., September 9, 2024. Last modified September 9, 2024. Accessed July 26, 2025. https://www.ign.com/articles/concord-victim-long-development-time-playstation.

MacDonald, Keza. “Sony’s Big-Budget Hero Shooter Concord Failed Spectacularly – Here’s Where It Went Wrong”. Guardian News & Media Limited, September 11,

2024. Last modified September 11, 2024. Accessed July 26, 2025. https://www.theguardian.com/games/article/2024/sep/11/sonys-hero-shooter-concord-failed-spectacularly-heres-where-it-went-wrong.

Nikolov, Saso. *Scrum Master Exam Vorbereitung*. North Charleston, SC: Createspace Independent Publishing Platform, 2017.

Olsen, Dan. *The Lean Product Playbook*. Nashville, TN: John Wiley & Sons, 2015.

Petrillo, Fabio, Marcelo Pimenta, Francisco Trindade, and Carlos Dietrich. "Houston, We Have a Problem...: A Survey of Actual Problems in Computer Games Development." In, 707–711, 2008.

Pichler, Roman, and Stefan Roock, eds. *Agile Entwicklungspraktiken Mit Scrum*. 1st ed.. Heidelberg, Germany: dpunkt, 2011.

Reuben, Nic. "Paradox Celebrate 10 Years of Cities: Skylines with a Roadmap of Anniversary Goodies". Rock Paper Shotgun, March 10, 2025. Last modified March 10, 2025. Accessed August 3, 2025. https://www.rockpapershotgun.com/paradox-celebrate-10-years-of-cities-skylines-with-a-roadmap-of-anniversary-goodies.

Schwaber, Ken. *Agile Project Management with Scrum*. Redmond, WA: Microsoft Press, 2004.

Schwaber, Ken, and Jeff Sutherland. "The Scrum Guide". Last modified 2020. Accessed July 10, 2025. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf.

Scullion, Chris. "Sony President, COO and CFO Hiroki Totoki Has Shared His Views on Why Concord Was a Failure.". 1981 Media Ltd., November 8, 2024. Last modified November 8, 2024. Accessed July 26, 2025. https://www.videogameschronicle.com/news/sony-president-reflects-on-why-concord-failed/.

Steigerwald, R., and Luqi. *Rapid Software Prototyping*. Dudley Knox Library. Monterey, CA 93943: Naval Postgraduate School, 1990. https://hdl.handle.net/10945/43606.

Upcraft, Steve, and Richard Fletcher. "The Rapid Prototyping Technologies." *Assembly Automation* 23, no. 4 (2003): 318–330. https://doi.org/10.1108/01445150310698634.

Wikipedia. "2022–2025 Video Game Industry Layoffs". Last modified 2025. Accessed July 26, 2025. https://en.wikipedia.org/wiki/2022%E2%80%932025_video_game_industry_layoffs.