Andreas Punz, BSc

Analysis of Collaborative Game Development Networks

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. BSc Johanna Pirker

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl-Inf. Dr. Stefanie Lindstaedt

Graz, October 2019

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Abstract

Video games are a popular past time and spark the desire in many people to create such games themselves. However, creating a game is a complex endeavor, that needs a lot of knowledge, practice, and organization. For these reasons, it can be difficult or even intimidating for some people to get started with game development. Game jam events can serve as a first stepping stone for these people to get some practical experience. Game jams generally require no prior knowledge, but even experts can participate for the sheer joy of creation. The Global Game Jam is a particularly popular iteration of this event type, taking place all over the world since 2009.
In this thesis, we analyze data from the Global Game Jam event to learn more about the game development process at the event. Of particular interest for us are its participants, how they work together and what tools they use. We collected data from the official Global Game Jam website ranging from 2014 to 2018, including games, participants and their skills. Where applicable we also gathered data of Global Game Jam projects that are hosted on the GitHub platform. We used this data to create networks for further analysis. We discovered that depending on the skills that participants possess, the number of connections they form with other people can wildly differ, as well as the chances that they will participate at the Global Game Jam event again. We also observe how such repeated participation affects their connections. We further discuss the use of tools used at the event and which platform participates like to develop for. We show how usage has changed over time as well as pointing out regional differences. We further analyze how participants use GitHub to more effectively work on their project during the event and to share it with the world afterward. In regards to GitHub, we discovered that participants that host their project there possess more skills and are more involved with the Global Game Jam event than those that do not. Therefore, we developed a team recommender application for the Global Game Jam event that is based on GitHub teams.

Contents

List of Figures

1. Introduction

Video games have become an ever more popular entertainment medium in the last few decades. The video game market has according to Wijman (2019) and Watson (2018) been outpacing the film industry, and the gap is predicted to become even bigger. This, in turn, sparked the desire of many people to become game developers themselves. The development of any software program is a difficult task, but for games, there are additional factors to consider. Developing a video game is more than a pure engineering task. In addition to making sure to deliver a high-quality product from a technical point of view, a game is also supposed to entertain the player. This is not something that can be achieved by only relying on computer science. So while working on any software project presents a multitude of challenges, when developing a game some occur more often than in other types of projects, as described by Petrillo, Pimenta, Trindade, and Dietrich (2009). Common problems in software projects are issues with schedule, budget, and quality. By analyzing specialized literature of game developers that looked back on their projects, so-called postmortems, Petrillo et al. (2009) were able to point out some problems that occur more commonly in games. One example is feature creep, where during development more and more elements are added to the game. Problems with software tools were also often reported, as well as issues in the design phase before the actual development even started. These problems can either lead to crunch time, meaning that the developers have to work overtime to meet the designated deadline or to a costly delay of that deadline.
Considering all those potential problems, game development can be a daunting task for newcomers. Game jam events can play a part here to provide some experience for beginners, as argued by Cook, Smith, Thompson, Togelius, and Zook (2015) and others. Game jams are social gatherings where the goal is to develop a game in a limited amount of time. Game jams provide a relatively risk-free environment for people who want to try out

game development by sidestepping some of the aforementioned issues. The teams in a game jam are self-organized and can work on their dream game without outside influence. Game jams have predetermined themes that help with the idea finding process, and the known time limit helps to avoid feature creep. Participants are also encouraged to use any tools they like, which provides a valuable learning experience. That is not to say that game jams are only for game development beginners. They are addressed at everyone, from people that have no idea about the technical aspects of game development to veteran developers. This gathering of people with different backgrounds is one of the most significant aspects of game jams, potentially leading to new innovative game ideas. In that sense, game jams can be an experience that widens the horizon of the participants. All of this, however, should not distract from another important aspect of the event: developing a game can be plain fun. Therefore, it is critical to understand the processes that take place at these events and how the participants interact.

1.1. Goals and Objectives

In this thesis, we aim to get a better understanding of the qualities of game jam participants. To do so we utilized tools such as data analysis, social network analysis, and GitHub. We collected the results of five Global Game Jam events, starting with 2014 until 2018 to answer the following questions:

1. Are Global Game Jam participants with specific skills better connected to other participants?
2. Do Global Game Jam participants that attend multiple events stick with the same group or do they prefer to work with new people?
3. Which technologies and platforms are used in the Global Game Jam?
4. Are there people that prefer to work alone instead of in a group, and what are their characteristics?
5. How is GitHub used during the Global Game Jam?

Furthermore, we also provide a visualization dashboard of the gathered data as well as the implementation of a team recommender that is based on our analysis.

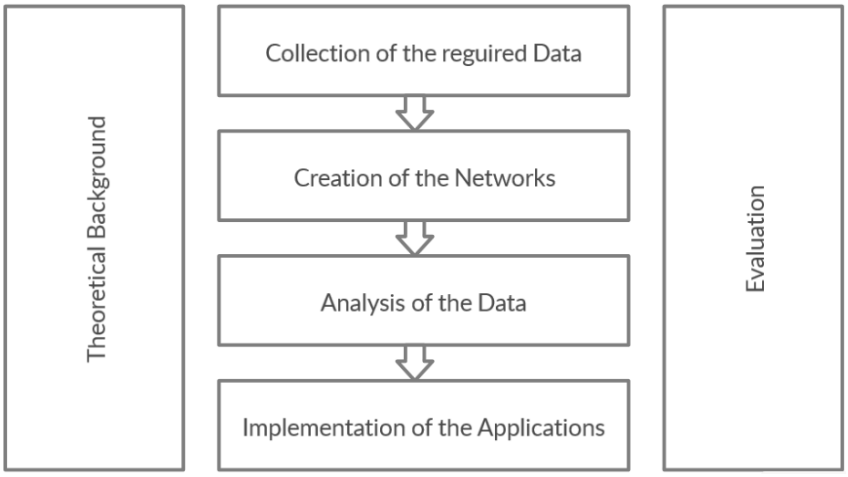

Figure 1.1.: Starting from the theoretical background, we go through the process of acquiring, structuring and analyzing the data, as well as the implementation of the applications. The final part is the evaluation.

1.2. Methodology and Structure

This thesis is divided into three parts. The first part provides the theoretical background of the discussed topics (Chapter 2). The second part is about the origin of the data (Chapter 3) as well as the networks that were created based on this data (Chapter 4). The last part is the analysis of the data (Chapter 5) and the description of the accompanying applications (Chapter 6). In Chapter 2, we define what constitutes a game jam event and introduce some popular iterations of this event type. We also discuss the challenges that participants of game jams face. We describe some more traditional software development models and discuss potential benefits of attending a game jam event.

In Chapter 3, we describe in detail the origin of the data. We show the profile pages of participants, games, and sites and which attributes were important for the analysis. We also show what data was acquired from GitHub. The limitations of this dataset are also discussed.

Chapter 4 deals with the networks that were built based on the collected data as well as the characteristics of the networks. Some network basics and

metrics are also described.

Chapter 5 is the analysis of the data that is essential to answering the research questions, as well as to the implementation of the team recommender. We take a look at who attends game jams, what tools they use and how they utilize GitHub.

In Chapter 6 the user interface and functionality of the visualization dashboard are described. For the team recommender, it is discussed how the recommending algorithm works.

Finally, in Chapter 7 we discuss which further topics could be explored in this environment, as well as problems that occurred during the work on this thesis.

2. Background and Related Work

Game jam events blend topics ranging from software development to interpersonal communication. So in this chapter, we take a look at game jams in general and some notable game jam events. We will also discuss some of the unique challenges that are attached to game jams from a participant's point of view. We will further describe some of the more commonly used software development methods. Finally, we take a look at other related work concerning the analysis of collaboration networks.

2.1. Game Development

The development of a game is at its very core, according to Bethke (2003), a software development like any other. Game developers often do not adhere to formal software development processes because in their view games are not just a product, but art. However, in mastering the fundamental software design principles game developers can make better use of their resources to create better games. The software development model that is used varies from team to team, but it often used to be a variation of the Waterfall Model according to Kanode and Haddad (2009). Over the years, agile methods have come to be preferred, such as Scrum, described in Chapter 2.3. The reason for this is that *"[p]rototyping and iterations are, perhaps, inescapable in video game development due to the indefinable nature of creating a "fun" game"* as Kanode and Haddad (2009) put it. The production of a game can be broken down into different parts, with each part presenting a different set of tasks. The following categorization of development roles is according to Bethke (2003) and Shyleno (2019).

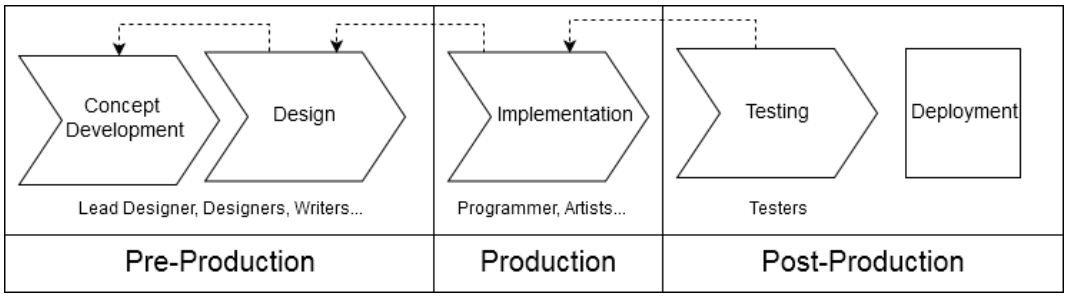

Figure 2.1.: Basic Game Development Process, adapted from Hirst (2008)

Basic Development Process

As can be seen in Figure 2.1 the development process of a game can be split into three major phases. At the beginning of the process is the pre-production phase. Juego Studios (2017) describes this as a crucial phase since it dictates where the entirety of development is headed. This stage includes building the base game design and story concepts, the selection of an engine and building a prototype. When the prototype is satisfactory, the game enters the production phase. Here, the coding, modeling, layout, animation and all other parts of the game are implemented. This results in an alpha version and later a beta version of the game. These versions are the basis for the evaluation of the testing team. After the development has wrapped up, the final product is reviewed in the post-production phase and made ready for deployment.

Design

At the start of every game is an idea or a vision. These come in some cases from the publisher or in other from the lead designer. The task of the lead designer, or director, is to come up with design specifications and communicate them with the development team, the publisher and other stakeholders. Of course, the lead designer cannot design every aspect of the game on their own, but they do have the final say of what does get implemented into the game and what does not. More designers work on specific parts of the game. Depending on the structure of the game they are

called level or mission designers. These designers can come from different backgrounds, from programmers who also work on the scripting of the missions or levels to artists who only lay out their ideas in text form.

Story And Writing

The writer is responsible for the game's story, and their workload and importance can greatly vary depending on the lead designer's vision. The writers have to work together with the other designers so ensure a consistent tone throughout the game. According to Bay (2018) the job of the writer changes significantly during the production of the game. In the early phases of development, they think up details about the game world, like locations and historical events, as well as the general plot. In later stages, they concentrate on the backstory of the characters and the dialogue.

Programming

The programmers are responsible to put the ideas and concepts of the design team into code. This begins with the lead programmers, who are usually the programmers with the most experience. They handle the most difficult tasks of the project and lay the groundwork for the other programmers. The lead programmers are also responsible for mentoring their subordinates and resolving conflicts between them, as explained by Marcelo Madril (2016). Some projects may also have technical directors, which are less concerned with programming and more with managing the programmers and the code they have produced. For pretty much any aspect of the game there is a dedicated programmer or even a team of programmers handling them. These aspects include game mechanics, artificial intelligence, user interface, audio, tools, network functionality and more.

Art

The art director's responsibility is to make sure that every piece of art that is made by the team is of not only of sufficient quality but also coherent with

one another. The art director is again the artist with the most experience on the team. The work of the concept artist primarily takes place before the game goes into full production. They sketch out the basic look of different elements of the theme and work in close correspondence with the art director. Only once a concept has the art director's approval, another artist is allowed to produce the actual art asset. The members of the art teams are also the ones that create the user interface. Shyleno (2019) adds that depending on if the game is in 2D or 3D, the expertise required by the artists varies. 3D games need artists for models and textures, as well as animators. 2D games, on the other hand, need sprite designers. But in recent times, event 2D games use 3D models as their base. Even though they all work in an artistic field, an understanding of the technological background is very helpful to understand the given limitations.

Audio

This part covers music, sound effects, and voice-overs. The number of composers depends on the game and what volume of pieces of music it should include. The sound effects are handled by audio engineers. They manipulate existing recordings of sound effects or create entirely new ones to fit the need of the game. They might also make use of royalty-free libraries. The person responsible for the voice acting is the voice-over director, which includes the hiring of the voice actors as well as directing their performance.

Quality Assurance

In smaller teams there is no dedicated testing staff, this part being covered by other team members instead. Larger teams, on the other hand, have their own QA department. The head of these departments is the QA lead, an experienced tester, familiar with QA principles as well as being passionate players themselves as stated by Shyleno (2019). They coordinate the main team and approve of their reports. The main team accompanies the game through every stage of development. New team members will often be rotated in as to make sure to have someone with the perspective of a new

player. Depending on the scope of the game there may be testing teams for individual parts of the game.

Management

As management builds the backbone of the development team, every role here has to be fulfilled with a great amount of care to make sure that the team can focus on making the best game possible. Mullich (2015) stress that "*[t]eamwork, communication, consensus, and trust are as essential components of a good game, as are design, art, audio and programming.*" The exact job titles for the people working here can vary from company to company, but they usually include the term "producer". First here are the line producers who schedule and take care of tasks that by themselves are not that challenging, but make up through the sheer amount of them. An example here is sending out updated design documents to the appropriate team members. Next, are the associate producer and the executive producer. The executive producer is responsible for strategic decisions, business negations and the like. They are also overseeing the general progress of the project. The associate produces report directly to the executive producer and have a lot of responsibilities, like task tracking and keeping the schedule up to date. Making sure that communication between team members goes smoothly also falls to them.

As can be seen, the production of a game is a lengthy, complicated process that needs a team of people with different experiences working together for a common goal. Game Jam events can provide a training ground for interested people, allowing them to prototype their ideas.

2.2. Game Jams

Game jams have become more and more popular in recent years, so we will try to define what the characteristics of such an event are as well as introducing some popular game jam formats.

2.2.1. Defining Game Jams

A game jam is a social event where individuals, in the following called jammers, gather to develop a game together in a relatively short amount of time. Kultima (2015) defines game jams as *"[a]n accelerated opportunistic game creation event where a game is created in a relatively short timeframe exploring given design constraint(s) and end results are shared publically"*.
Fowler, Lai, Studios, and Khosmood (2015) explain that there are several different philosophies for organizing a game jam event in the first place, like the regional affiliation, the use of certain technologies or dealing with particular social topics. Regardless of these differences, there are some common points between popular game jam events that Kultima (2015) describe in more detail based on literature research. Fundamentally, game jams are events or gatherings to create games. An important point here is that there are no prerequisite skills or knowledge that the participants, need to have. Fowler, Khosmood, and Arya (2013) say that anyone who can contribute in some way to the development of a game and is willing to put effort into it is welcome. Some restrictions regarding age and school affiliation may still apply, depending on the event. Game Jam events also usually have a set time limit for the development of the game. 48 hours is a popular amount of time and well-suited for events that take place on a weekend. The short time frame is often the major limiting factor in the development of the games, Musil, Schweda, Winkler, and Biffl (2010) state. A more controversial point among the different events is if they are considering themselves as a competition. The events that do not hold a competition base this on the idea as Goddard, Byrne, and Mueller (2014) argue that failure should be embraced, and competition would distract from that. Regardless of that, however, the games that the jammers developed are presented at the event and sometimes even made public by the jammers.
Teamwork is also a major point in game jam events. It usually not permitted to form teams before the event starts. Game jams often employ a theme for the event and the games, which gives jammers a starting point for which direction their game should go in. All the constraints that are placed on the jammers during the event, like the time limit, the team building, and the theme have the goal of encouraging jammers to get to the core part of development as quickly as possible instead of meandering with endless

brainstorming sessions. The constraints that are part of a game jam event, in combination with the lack of any commercial concerns, encourage new and bold game design ideas, Deen et al. (2014) argue. As Kultima (2015) put it, *"[a] game jam is accelerated through the constraints [...]"*. It should be stressed that these points are not a hard definition, and more like a guideline to which events could be considered a game jam since the specifics can vary vastly from location to location.

2.2.2. Popular Game Jam Formats

All around the world, different formats of game jams take place, with their own rules and peculiarities. We will describe some of them in more detail in this section.

Global Game Jam

The Global Game Jam[1] is an event that takes place every year on a global scale, first starting in 2009. The event is organized by a non-profit organization, Global Game Jam Inc. and the Global Game Jam executive committee. Each year, there is a specific theme for the event that the jammers are supposed to develop a game around in 48 hours. This theme is the same for all locations and a secret before the event starts and is only announced on the Friday the event starts, 5 PM local time. For example, the theme for the 2014 event was *"We don't see things as they are, we see them as we are"*, Global Game Jam (2014). However, due to time zone differences jammers are not supposed to publicize the theme before the Hawaiian event begins. While it is necessary to register online to participate in the event, it is not allowed to work on projects online. Jammers have to be present on their registered jam sites and develop their games there. Team building is done on the jam sites as well, but jammers can also work on their own if they choose to do so. There are no requirements in terms of used technologies and targeted platforms either, so jammers are encouraged to get creative. While the events are not a competition, there may be a presentation on the

[1] https://globalgamejam.org/

jam site that features games selected by a jury.
The Global Game Jam event is open to everyone over the age of 18 and no prior knowledge of game development skills is necessary. Non-Digital games like board games are welcome as well. So while most of the jammers are amateurs that primarily develop games for the inherent fun of it or to hone their skills, some of the games developed during the event have gone on sale on platforms such as Steam[2]. A few notables examples can be found here[3]. Over the years, several prestigious companies have come to sponsor the Global Game Jam, like Epic Game, GitHub or Valve, to name a few.

Indie Game Jam

The Indie Game Jam[4] was an event that took place four times from 2002 to 2005. These events had a duration of four days. During that time, a group of game developers tried to make as many games as they could. Particularly noteworthy about this event is that the choice of engine was not up to the developers. Instead, they had to use a custom-made engine that was specifically developed for the event. This was done according to Hecker, Ulrich, and Hall (2004) to support the event's main goal to "*encourage experimentation and innovation in the game industry*".

Nordic Game Jam

The Nordic Game Jam event[5] happens annually since 2006 in Copenhagen. In addition to the game jam, there are other events on the days preceding it, as can be seen on Nordic Game Jam (2019). Namely the StartUp day, which includes panels and workshops from game publishers where attendees can learn how to get started in professional game development. There is also the Dev Day, where attendees can listen to talks of industry insiders about new game design concepts or development tools. After those events, the main event starts.

[2]https://store.steampowered.com
[3]https://globalgamejam.org/ggj-success-stories
[4]http://www.indiegamejam.com/
[5]http://nordicgamejam.com/index.html

Like many other Game Jams, the Nordic Game Jam also has a time limit of 48 hours for the teams to finish their project. Participants have to be physically present on the jam site, taking part in the event over the Internet is not permitted. There are not any kind of limitations regarding the content or the type of game, so for example card games are allowed as well. On Sunday after the projects are finished, a public vote is held to determine ten finalists that give a presentation on the main stage. The winner of the event is elected of those games again by a public vote. A ticket to the Nordic Game Jame also includes six vegetarian meals plus free toast and coffee.

Slavic Game Jam

The Slavic Game Jam[6] is another annual game jam event that took place four times so far. It is located in Warsaw, Poland and organized by KTNG Polygon. Besides the yearly game jam, they also organize weekly workshops and lectures. The main event is a 48-hour game jam, but other activities are also part of the event. In 2018, the Slavic Game Jam had over 200 participants, according to Slavic Game Jam (2018).

Ludum Dare

Ludum Dare[7] is an online community that organizes the Ludum Dare game jam event. It was founded in April of 2002 and began with the challenge of creating a game from scratch within 24 hours. Ludum Dare events typically take place in April and October. The games are also developed to a certain theme, which is voted on by the community a few weeks before the event takes place.
As can be seen on Ludum Dare (2017), there are two different categories for games at this event, the Jam and the Compo. The Jam is similar to other game jam events, in that within a time limit of 72 hours a game is developed within a team or alone. The use of preexisting base code or third-party assets is allowed, and there are no restrictions regarding the used tools. At the

[6]https://slavicgamejam.org/
[7]https://ldjam.com/

last event, the Ludum Dare 44, 71.59% of the submitted games were for the category Jam. The second category, the Compo, differs by every participant having to work on their own instead of in a team and only having a time limit of 48 hours. Also, any assets that are used in the game have to be created from scratch and participants are expected to share the source code of their game. After the event, everyone who submitted a game is allowed to rate as many of the other developed games as they like. The games are rated in different categories like "Innovation" or "Graphics". Games in the Compo category are also more harshly judged than those in the Jam category. Notably, everyone who rates a game also receives a "Coolness" rating that depends on how many games they have rated themselves. The rating of people with a higher "Coolness" rating is prioritized. However, there are no prizes for the highest-rated games, the competition is just for fun.

48hr Game Making Challenge

Beginning in 2007, the 48hr Game Making Challenge[8] as described byWarton (2015), is a game jam event that is usually held around October in Brisbane, Australia. It is organized by the Brisbane International Game Developers Association. As the name implies the time limit for this event is 48 hours. The teams are grouped into three different leagues, depending on their skill level. The first category is Indies, which is for students and hobby developers that are new to game development. Next is Indie Pro, which is for jammers that already have some experience in game development, like from attending another game jam for example. They also share space with the Indie league. Pro is the final category, and it is for full-time professional game developers. The teams can decide for themselves in which league they feel most comfortable in. All entries are judged at the end of the event by senior staff of local game development studios.

[8] http://www.igdabrisbane.org/48hr-challenge/

	GGJ	IGJ	NGJ	SGJ	LD	48hrGMC
Established	2009	2002	2006	2015	2002	2007
Ongoing	Yes	No	Yes	Yes	Yes	Yes
Frequency	Annual	Annual	Annual	Annual	Biannual	Annual
Length (hours)	48	96	48	48	72/48	48
Online Participation	No	No	No	No	Yes	No
Theme	Yes	No	Yes	Yes	Yes	Yes
Choice of Tools	Free	Custom*	Free	Free	Free	Free

Table 2.1.: Feature Comparison of Different Game Jam Events. GGJ: Global Game Jam. IGJ: Indie Game Jam. NGJ: Nordic Game Jam. SGJ: Slavic Game Jam. LD: Ludum Dare. 48hrGMC: 48hr Game Making Challenge. *"Custom" refers to the engine that was specifically made for the event.

2.2.3. Challenges of Game Jams

Participating in a game jam poses numerous challenges to the jammers, some of which will be discussed here. One of the first and biggest challenges takes place before the development of the project even begins, the team building. While the exact details of the process of team building vary from event to event, the time frame to form a team is relatively short, especially considering that in many cases the team members do not know each other beforehand. This makes it difficult to know how well the team will work together based on their technical, but also their social skills. Some skills are also more common among jammers then others, which might mean for some of them that they get the chance to work on multiple projects during a single game jam, as observed by Pirker and Voll (2015).
Another challenge is the choice of development tools. While each jammer has their own set of tools that they are most comfortable with, for the sake of cooperation the team has to agree on a compatible set of tools. Combining the work of all jammers to a single working project is also not always a simple task, especially in larger teams. Tools like Subversion[9] or GitHub[10] can be used to make it easier for the jammers to manage their projects.
In most game jam events, the games have to adhere to a certain theme, which will only be announced shortly before the event starts. This, of course, makes it hard to prepare any meaningful work in advance, which challenges jammers to come up with new game ideas on the spot. One major challenge a game jam poses is the time limit, typically around 48 hours, according to Borg, Garousi, Mahmoud, Olsson, and Stalberg (2019). Building any type of software from scratch within such a short amount of time is a difficult task, especially when some members of the team have little to no experience in game development.
To summarize, the challenges that a game jam event poses are:

- Team building
- Strict time limit
- Choice of tools
- Managing the collaboration
- Predetermined theme

[9]https://subversion.apache.org/
[10]https://github.com/

Participating in game jam events is still a worthwhile endeavor for the opportunities they provide like:

- Learning new skills and tools
- Interdisciplinary cooperation
- Risk-free environment
- Putting theoretical skills into practice
- Networking opportunities

These points and more are discussed in detail in Chapter 2.2.5. There we will discuss research that deals with game jam events in terms of development processes during the event and what opportunities they provide to participants. We will also examine the effects of GitHub on the software development process.

2.2.4. Cooperation in Game Jams

Musil et al. (2010) analyzed how the development of projects during game jams differs from other more traditional models by characterizing three steps. Firstly, in terms of interaction design parameters, they qualify game jams as taking a pragmatic account, meaning that the design of the product is reflective of the collective experience of the team. The end product is not the result of a structured design process, but rather the result of an ongoing conversation and problem solving between the team members. Secondly, as far as Development Process Structures are concerned, Musil et al. (2010) conclude that game jams do not fit with any other known software engineering process. Therefore, they compare it to a concept that applies to more than just software development, the basic system engineering process (SEP), as described by Lightsey (2001) (Figure 2.2). The development during a game jam is similar to that of the SEP in that it is a *"comprehensive, iterative and recursive problem solving process"*. The third and last step compares game jams with existing collaboration concepts and in what way they differ. Some comparable collaboration concepts they found are MIT's Invention Lab or by describing game jams as "design games". Beyond these three steps, Musil et al. (2010) go into more detail and identified eight elements (Figure 2.3) that describe the game jam design process. These are:

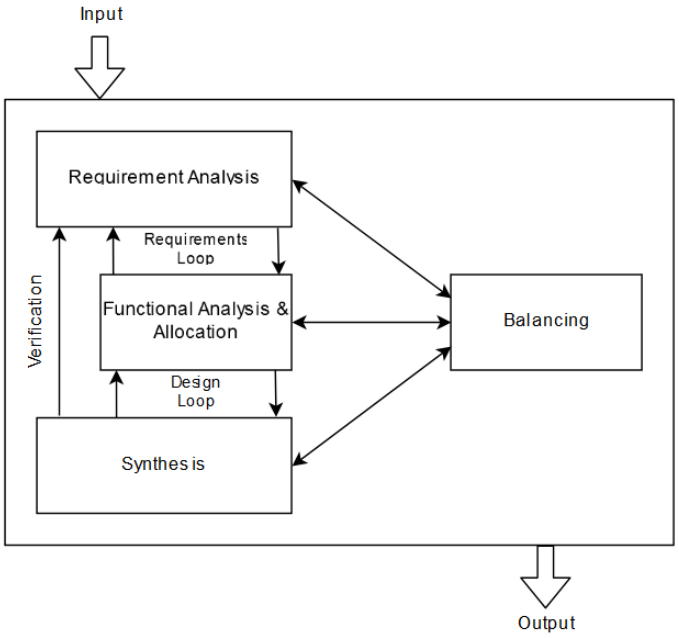

Figure 2.2.: The system engineering process, adapted from Lightsey (2001)

1. **New Product Development:** The games that are developed during a game jam can be described as the result of a short term new product development. Experiences of the team during development have an immediate effect on the end product.
2. **Participatory Design:** During a game jam there are no fixed team roles, and they also present a relatively risk-free environment for anyone regardless of prior experience to present their ideas to contribute to the project.
3. **Lightweight Construction:** The focus of development during a game jam is on rapid prototyping and to rely on rough simulation of more complex aspects of the project.
4. **Product Value-Focused:** Since there is a strict time limit in most game jams, the team has to concentrate on a set of core features that they want to integrate. It also has the effect of being more open to the idea of dropping features that do not work out.
5. **Rapid Experience Prototyping:** Game jam projects can be described as experience prototypes. That means that they are more focused on

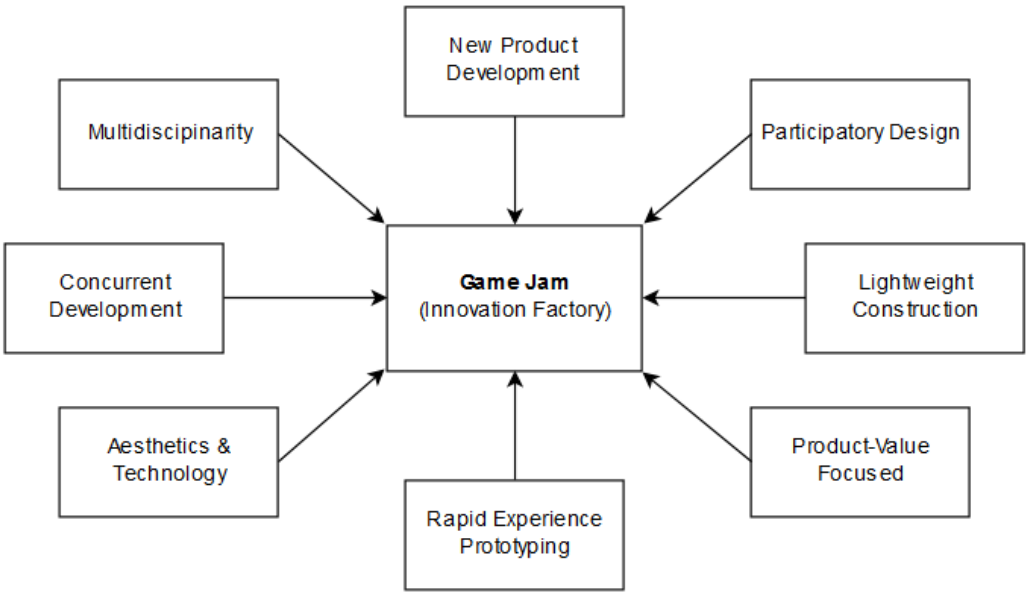

Figure 2.3.: The eight elements of a game jam, adapted from Musil, Schweda, Winkler, and Biffl (2010)

the aesthetic integrity of the product rather than its technical qualities.

6. **Aesthetics and Technology:** In normal software product development it is important to consider factors such as scalability or reusability. For game jam projects, however, only the end product matters.
7. **Concurrent Development:** Game Jam projects are typically worked on by small teams with tools and systems of their choice. So the end product *"can be regarded as a collection of solution sets that approximate a given problem domain"*, Musil et al. (2010) state.
8. **Multidisciplinarity:** The teams that participate in a game jam ideally consist of members with background knowledge from different fields. This combination can be a fertile ground for new and innovative ideas, leaving the beaten paths behind.

In conclusion, Musil et al. (2010) say that game jams are a *"collaborative, multidisciplinary, concurrent set-based development approach for experience prototyping"*. This approach may also be viable for non-gaming software products.

2.2.5. Analysis of Game Jams

Some aspects of game jams have already been scrutinized in various studies, like their advantages and challenges. In the following, we will show some of that research.

Benefits of Game Jam Participation

Game Jams can be an excellent opportunity to put one's abilities into practice and further honing them. Participating in such an event can also be used to learn new skills. Learning is one of the main motivations for attending a game jam event, as Fowler, Khosmood, Arya, and Lai (2013) asses. In terms of acquiring new skills, they show in their work what impact taking part in a Global Game Jam can have on the participant's skills. To do so, they did a survey during the Global Game Jam 2013 with over 1,200 participants. Around 60% of the study's participants answered that they have had acquired new skills during the event. The event was also an opportunity for participants to learn new tools, as 65% stated that they have made new experiences with tools like game engines, art tools, or sound tools. Also, almost all participants reported that their skills improved at least a little due to their participation in the Global Game Jam.
In a similar vein, Pirker, Economou, and Gütl (2016) conducted a survey with twelve Austrian computer science students and ten students from the United Kingdom from various fields, who took part in a game jam event. 71.43% of all participants agreed that the skills that they have acquired during the game jam would be useful for their later professional life, like for example getting more experience with programming languages like C# or getting their first hands-on with completely new tools like Blender. Almost all of them would consider taking part again in an event like this.
Cook et al. (2015) also argue for the benefits that participation in game jam events provide, particularly in terms of interdisciplinary cooperation. When developing a game, people who are experts in certain fields have to make sure that their systems and concepts efficiently work together with those of others to result in a functioning game. Game jams can also spark the creativity of students, since they are a relatively risk-free environment, without the need for detailed project plans that the participants may be

used to in their normal work environment.
B. Law and McDonald (2015) discuss how game jams, in general, are a good opportunity for undergrad students to practically apply the skills that they have learned during their coursework. Another benefit students get from participating in a game jam event is learning how to work with people whose skill set differs from their own. B. Law and McDonald (2015) further mention that tools like GitHub[11] are becoming more important to efficiently work on a project. There are even considerations from Deen et al. (2014) to adopt the approach of game jam events from game development to research.
To determine the reasons why jammers participate in a game jam in the first place, Wearn and McDonald (2016) conducted two surveys, one with Staffordshire University alumni and another with the Glasgow Caledonian University. In both surveys, ”Working with friends” and ”For fun” were the two most popular reasons to take part in the game jam. ”Trying out new ideas” and ”Networking” were mentioned as well in both polls. Wearn and McDonald (2016) recommend to conduct similar surveys before the start of other Global Game Jam events to get a better understanding of the motivations of the jammers and in turn improve the event.
A similar study about the reasons why jammers want to participate in a game jam event was done by Reng, Schoenau-Fog, and Kofoed (2013). They conducted two surveys, one before the start of the event, in this case, the Nordic Game Jam, and one after it was finished. Additionally, they also did some interviews with the jammers during the game jam. Of the total of 470 jammers that visited the Nordic Game Jam that year 110 participated in the pre-event survey and 74 in the post-event survey. As part of the pre-event survey, the jammers were asked if they would hope to develop a game in a group, which 95% answered positively. The jammers were further asked what they hope their tasks will be during the game jam. The most popular answers here were programming and designing, with 59% and 65% respectively. In the post-event survey, jammers were asked what they actually did during the event, and only 49% spent time programming. Another part of the survey was for jammers to estimate their game development skills on a scale from zero to ten, with zero meaning ”no experience” and ten meaning ”expert”. As can be seen in Figure 2.4, the game jam was a humbling experience for many jammers. However, 86% of

[11] https://github.com/

the jammers also said that they were at least to some extent motivated by the event to learn more about game development. Reng et al. (2013) conclude with stating that game jams can be a good first step into professional game development and an opportunity to build relationships with like-minded people.

Game Jam Entry Barriers

Of course, to reap any of the mentioned benefits of game jams people have to attend them first. Meriläinen and Aurava (2018) interviewed first-time game jam attendees about the reasons for the internal barriers they had to overcome to participate in the event. Meriläinen and Aurava (2018) identified four themes. The first theme is about social support and apprehension about the jam community. The interviewees cited fear about not being able to work with their friends and the other attendees being far more experienced in game development than they are. The second theme discussed the perceived technical competence of the interviewees. One interviewee explicitly mentioned that they were hesitant to go to a game jam event before because they assumed they were not able to contribute. Another interview participant said that although they are confident in their skills regarding writing, having no programming experience led them to conclude that game jams are not for them. The third theme talks about personal characteristics, mainly a lack of self-confidence. The interviewees also viewed game jams events as more of a competition. For the fourth theme, the interviewees gave their suggestions to lower the barriers of entry. They mentioned that a preliminary event for first-time attendees would be helpful. Discussing how people that are not into programming can help during a game's creation was deemed as especially important.

Development Processes at Game Jams

Stepping away from the benefits and personal challenges inherent to attending a game jam, Zook and Riedl (2013) took a look at the actual development process of the participants of the Global Game Jam 2013. Zook and Riedl

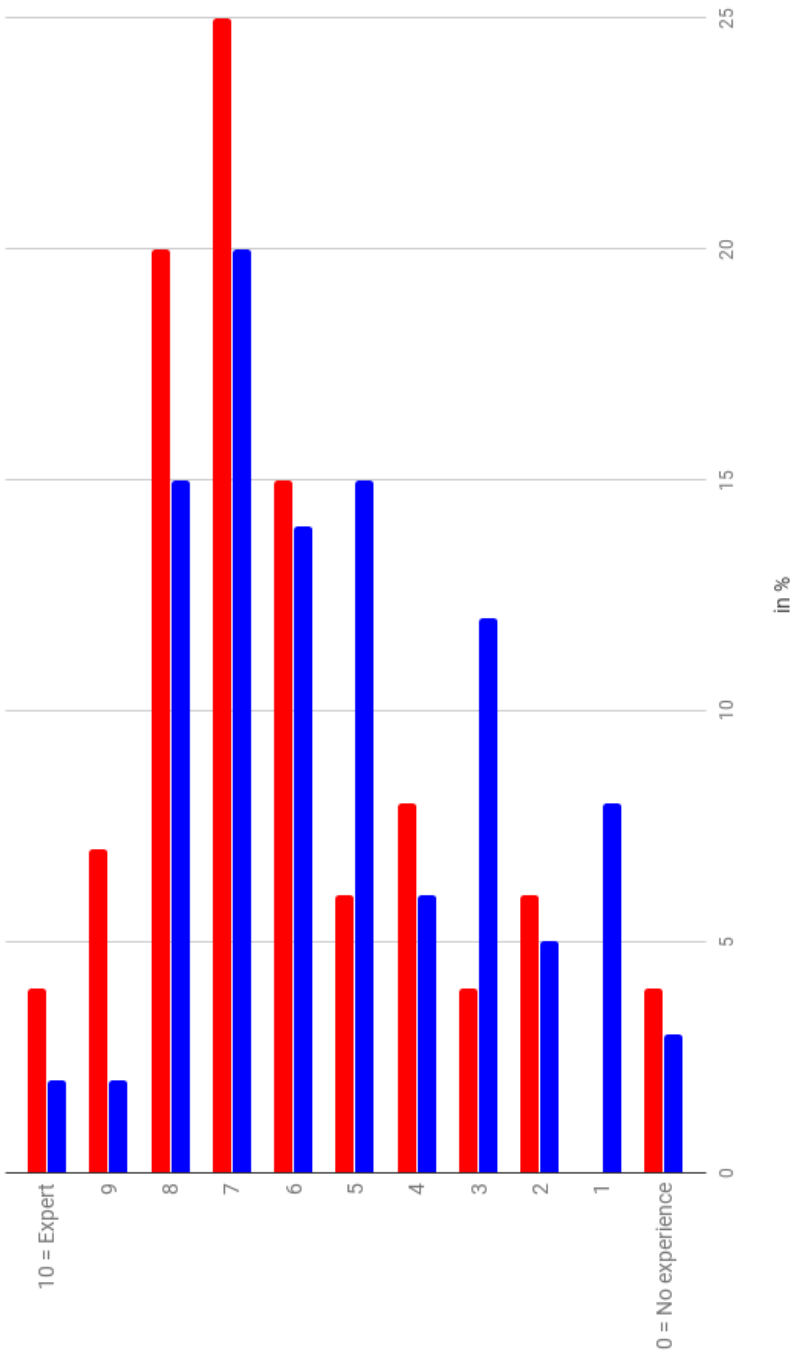

Figure 2.4.: Jammer's estimation of their game development expertise from Reng, Schoenau-Fog, and Kofoed (2013). Red bars stand for their estimation before the event, blue bars for afterward.

(2013) did this in the form of a post-event online survey. The survey contained nine open-ended questions. They began by asking the jammers what inspired the initial idea to their game. Unsurprisingly, the theme that was attached to the game jam was the most popular reason. Wanting to implement a specific mechanic, other video games, and the desire to develop a game of a particular genre were also named often. They further asked about the participant's goal regarding the game jam, divided into personal goals, player-oriented goals, and system level goals. Personal goals are about what the jammers themselves would like to achieve during the event. Here, the by far the most often given answer was making and completing a game. Player-oriented goals refer to what they hope the players of their games would feel about it. Besides the hope of their game being enjoyable, many jammers hoped that their players would learn something new from their game or become aware of certain issues. The system level goals are about what technical specifications the jammers want their games to fulfill. Recreating other games and implementing certain mechanics were important to jammers here.
In regards to the actual development process, Zook and Riedl (2013) asked if the jammers did any prototyping. Less than half replied positively, citing time issues or that they still consider their finished project as a prototype. Most of the jammers that did do prototypes employed engine prototyping, but some used paper prototyping instead. Another question was about the problems the jammers encountered while making their game as seen in Figure 2.5. Programming related issues and learning new tools were by far the most often encountered problems. Other encountered issues were with time, the group or with the art of the game. Zook and Riedl (2013) categorized the approach jammers took in regards to feature implementation in three groups: First, those who have many ideas at the start of development and then see what they can fit it in. About half of the jammers were fell into this group. Second, are those who have only a faint idea of the finished product at the start and build up more as development progresses. This group represents the smallest amount of jammers. Third, those that start with a core idea of the project and then iterate on that idea through player feedback and testing. When asked about how the design of their project changed during development, the most common answer from jammers was that features were cut. However, the second most common answer was that nothing changed at all.

Borg et al. (2019) also took a look at the development process during a game jam. They surveyed 198 participants of the Global Game Jam 2017. The participants were first asked about how they came up with game ideas at the beginning of the event. Brainstorming was the most common answer here, with 76% of the participants employing this technique. Prototyping was not used to find ideas, but more so to test out if ideas were technically feasible. Many participants also added that they kept the idea finding phase relatively short due to the 48-hour time limit. The next question was about how the teams handled changes from the initial plans that happened during development. The most common answer here was that all changes were discussed with all group members until a consensus was reached. The small group size of game jam teams and the fact that they are all working nearby each other makes this a very viable approach. The next topic in the survey was about quality assurance (QA). Borg et al. (2019) asked about the QA processes that were employed during development as well as on the finished product. During development, regular discussions among team members were again the most popular answer, followed by internal playtesting. Dedicated members for QA, external playtesting or simply doing no QA at all were comparatively rare. On the finished product, many participants liked to do internal playtesting for QA purposes. For the next topic, the participants were asked which software development practices they used during the event. The results can be seen in Figure 2.8. Borg et al. (2019) conclude by stating that communication is a crucial part when participating in a game jam event. The development environment of a game jam is also similar to that of a software startup, citing the time pressure and the small size of the development teams.

2.3. Software Development Methods

As software projects became bigger in scope, during the years new methods to manage the development process emerged. Software development models describe the general flow of development, from the initial stages of assessing the requirements to implementation and maintenance. Knowing them is helpful for any developer, be it for games or other projects, so we will discuss a few of them in the following.

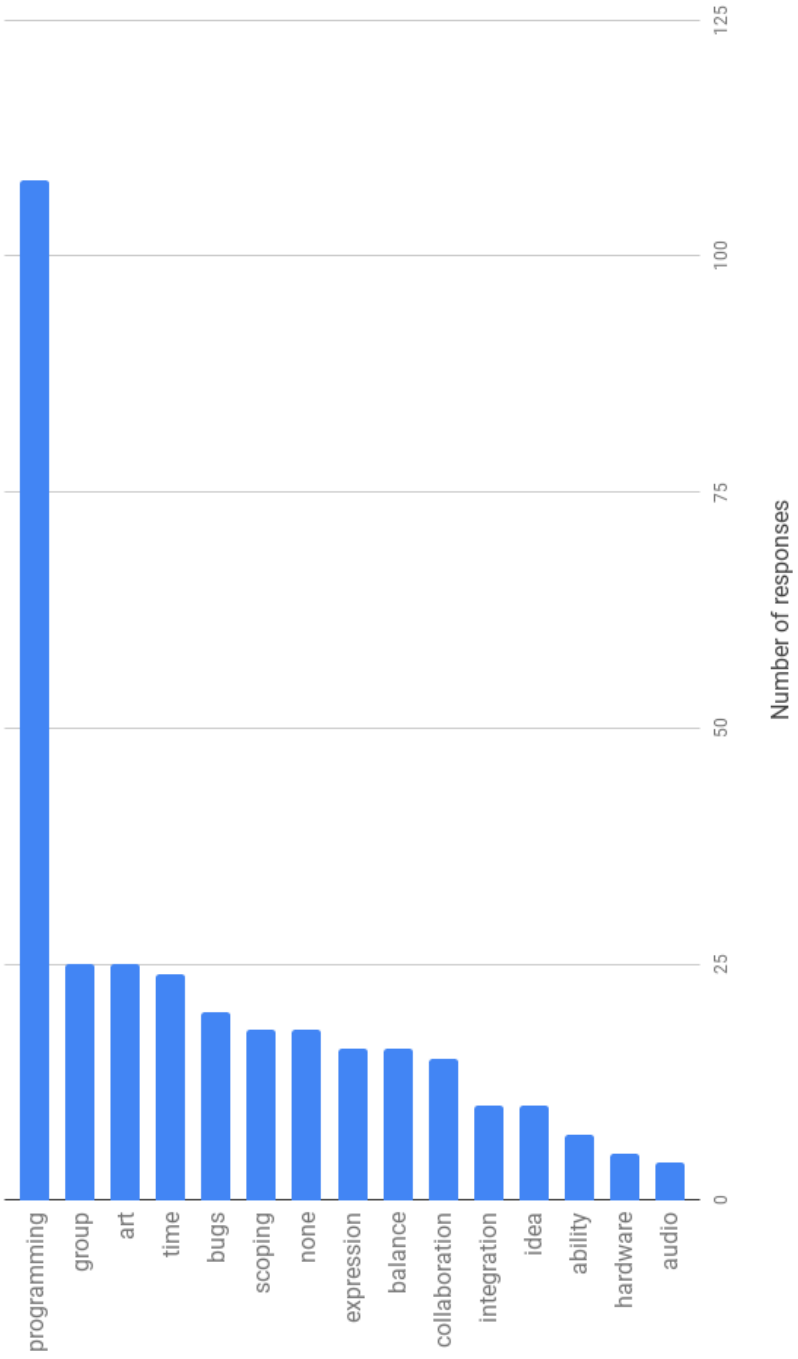

Figure 2.5.: Problem sources during development, adapted from Zook and Riedl (2013).

Waterfall Model

A development model described by Royce (1987), the Waterfall Model is divided into distinct steps, seen in Figure 2.6:

- Requirements: In the first step, the development team tries to assess the needs of the user and put these into a requirement specification. This step can be optionally broken up into System Requirements and Software Requirements.
- Analysis: In this phase, the previously defined requirements are further refined and potential challenges of development are investigated.
- Design: The development team assesses which hardware and software systems should be utilized to fulfill the requirements in the best possible way.
- Coding: The actual development of the project. The better the previous phases have been executed, the smoother the development will be.
- Testing: A thorough testing of the software is required to find bugs, but also if the program works according to the specifications.
- Operations: In this phase, the software is implemented into its working environment so that it can be operated by the user. Maintenance and support also may be a part of this phase.

The advantages of the Waterfall Model are that it is easy to understand and implement. Every step of the process also brings with it a lot of documentation. However, it takes quite a while before an actual functioning program exists, so mistakes made in the first phases of the development might not be noticed until much later. As previously mentioned by Kanode and Haddad (2009), variations of the Waterfall Model were commonly used by game development studios, but have since become outdated. Ferrara (2018) says that one of the reasons for this is that for most games the initial pitch has little in common with how the final product turns out, making the Waterfall Model useless. Therefore, it has since been largely replaced by agile methods, two of which are discussed below.

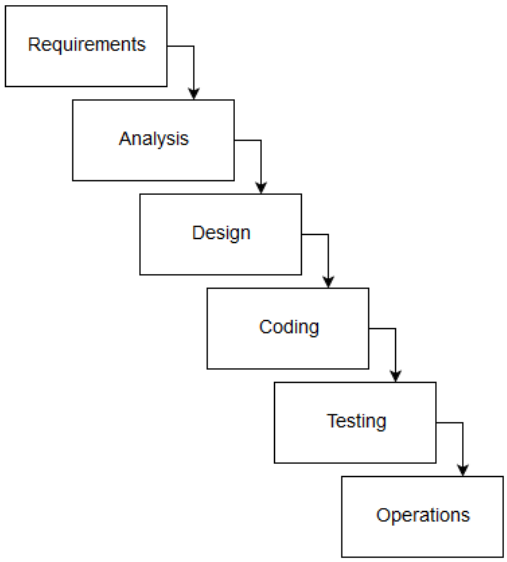

Figure 2.6.: Basic flow of the Waterfall Model, adapted from Oxagile (2014)

Rapid Application Development

Rapid Application Development is an approach that was first introduced by Martin (1991). The goal of the model is to deliver high-quality systems in a short amount of time as well as keeping the costs low. Beynon-Davies, Carne, Mackay, and Tudhope (1999) conducted seven case studies about the use of Rapid Application Development, and they noted that in some cases the teams ignored major parts of the model. They also explained the following components that are common to the model:

- **Joint Application Design:** The teams are relatively small, consisting of four to eight members. These teams do not only consist of developers but users as well. Both groups are expected to be well versed in regards to business and social skills. Workshops, meetings, and dinners are an important part of the team-building at the beginning of the project. Further, teams normally produce a set of business requirements, such as deliverable phases, in a few days.
- **Rapidity of Development:** The scope of projects is rather limited, usually only lasting about two to six months. The proposed reason for

keeping the project length to six months at maximum is that otherwise, the business developments may change too much.

- **Clean Rooms:** The workshops should take place in "clean rooms", away from the busyness of the rest of the company so that the team can concentrate on the problem-solving process.
- **Time Boxing:** For all phases of development, as part of managing the project, there have to be set deadlines. These are also called timeboxes. If problems start to arise in development, the product is not delayed. Instead, the requirements of a phase are reduced to still fit in the timebox.
- **Incremental Prototyping:** An essential component of the model is that the developers are quickly creating a working prototype based on initial research. This prototype is then discussed with the users in the team about what changes and improvements can be made. This process is repeated at least three times and corresponds to the prototype cycle in Figure 2.7.
- **Rapid Development Tools:** To enable the repeated prototyping cycles, development tools that make it possible to do quick changes to the project are needed.
- **Highly interactive, low complexity projects:** The Rapid Development Model is most commonly used in projects that are computationally simple but require a high amount of interactivity.

Due to prototypes being essential to the development process, Rapid Application Development is well suited for game development. For example, Sanneblad and Holmquist (2003) developed a platform for mobile game development that is based on Rapid Application Development.

Scrum

Scrum is an agile development model first proposed by Takeuchi and Nonaka (1986). Scrum teams typically consist of three to nine members, plus the Scrum Master and the Product Owner, Schwaber and Sutherland (2011) explain. The task of the Product Owner is to represent the customer and communicate their needs to the development team to get the best possible product value. The Scrum Master is responsible to create a beneficial work

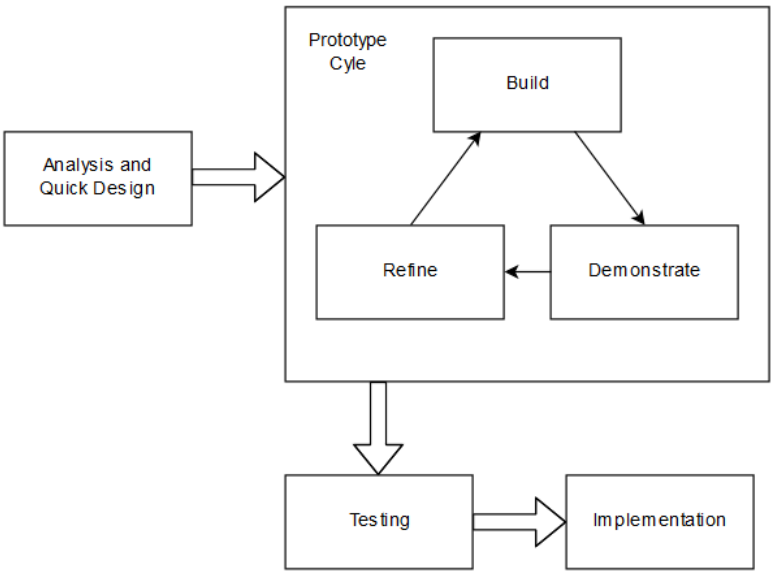

Figure 2.7.: The Rapid Application Development model, adapted from Ghahrai (2018)

environment, coaching the team members and making sure that the general Scrum framework is followed. The development consists of sprints, which can have a length of a week up to a month. During a sprint, new features are planned, implemented and tested. At the end of a sprint, there is always a functioning product. Scrum has six defining characteristics set by Takeuchi and Nonaka (1986):

- **Built-in Instability:** The project team is often given an extremely challenging project goal to create tension within the development team. The idea here is that this tension leads to creativity.
- **Self-organizing Project Teams:** There are three qualities to self-organizing project teams. Firstly, they have autonomy. That means that the influence of management on the actual development is limited, with the team being free to find its own approach to development. Secondly, there is self-transcendence. Based on the initial goals set by management, the team is supposed to find their own goals and refine them during development. Finally, there is cross-fertilization. If the team consists of members with different technical backgrounds, skills or personalities, this may lead to an automatic learning process from one

another.

- **Overlapping Development Phases:** In contrast to the Waterfall Model, in Scrum the development phases overlap so that the team can better adapt to changes in the project's goals or the market conditions. Overlapping phases lead to a quicker speed of development and provide more flexibility, but also requires a lot of communication within the team.
- **Multilearning:** Team members are encouraged, or sometimes even pressured, to learn, broaden their knowledge and acquire new skills, especially in fields besides their primary expertise.
- **Subtle Control:** While management largely stays away from actually controlling development, it can still steer the direction the project is going in by for example selecting the right people as project leaders, by establishing a reward system or by encouraging team members to get into contact with customers.
- **Organizational Transfer of Learning:** Knowledge is transferred by placing key individuals of previous projects in leading roles in subsequent projects. Also, the newly gained experiences during a project can lead to new standard practices for companies.

There are however some disadvantages to this model. Firstly, it can lead to a lot of overtime for the development team, to the excess of 100 hours or more per month during the peak of the development. Also, the model might be ill-suited for projects that "*require a revolutionary innovation*" as Takeuchi and Nonaka (1986) say or for projects of a bigger scope.
In terms of game development, Cohn (2008) argue that Scrum can help to reduce wasted efforts, minimize the crunch periods and being able to find the "fun" of the game quicker. Asuncion, Socha, Sung, Berfield, and Gregory (2011) also state that during the development of their games following the principles of Scrum helped them avoid difficulties that often occur during game development like setting unrealistic release dates.

2.3.1. Collaboration in Software Engineering

The successful completion of any software development projects, game jam projects included, depends on one hand on technical aspects, like for

example the tools used, and on the other on social aspects, meaning how the team members work together. Ortu et al. (2015) used the Jira Tracking system[12] to extract and analyze developer discussions of four open-source projects in the form of comments. They found that in addition to technical information these comments also contain numerous instances of developers communicating their feelings about the project and their work on it. They also identified several potential research opportunities for example if there is a connection between the software metrics and the emotions of the developers.
A. Law and Charron (2005) looked at how agile practices affect factors such as knowledge sharing or motivations. For this, they looked at a company with two distinct programming teams. For example, pair programming could be used to encourage knowledge-sharing, but if the pairs never change up then the knowledge stays within the pair instead of being made available to the whole team. Pair programming can also harm motivations if the personalities of the programmers do not match. A factor that helped improve social relationships are the daily meetings. In conclusion A. Law and Charron (2005) say that *"[t]here is no silver bullet"* that can be used to always ensure that a project will be successful.
A work by van Kelle, Visser, Plaat, and van der Wijst (2015) tried to construct a conceptual model to assess the value of communication and leadership related factors in an agile software development environment. In building their model, they identified three factors in particular that influenced the success of a project. Firstly, transformational leadership, referring to an adaptive type of leadership centered around motivation and social interactions. Secondly, value congruence which refers to how closely aligned the team members are in terms of what the goals of the project are. The third factor they determined was the degree of agility, meaning how much the development process is aligned with the ideals of agile development. Also, van Kelle et al. (2015) concluded that the size of the development team does not matter.

[12]https://www.atlassian.com/software/jira

2.3.2. Version Control Systems

Version control systems are essential in today's software development environment, as argued by De Alwis and Sillito (2009). Having a well-functioning version control system makes it easier for multiple people to contribute to a project, implement new features as well as reviewing and testing the program. There are two different types of version control systems, centralized ones, and decentralized ones. Centralized systems are characterized by only having a single canonical repository. So every developer has to work against this single repository by doing a so-called checkout. This checkout provides a snapshot of what the repository looked like at the time. Every time something new is committed, the master repository changes. To better accommodate multiple developers working on the same files, developers can create branches for them to work on, which are later merged back into the main branch. Subversion is a popular example of a centralized version control system. In decentralized version control systems, every developer has a first-class repository, containing the entire commit history. That way there is no forced master branch, rather it is chosen by convention. Decentralized version control systems are well suited to distributed development environments. Git is an example of such a system.

2.3.3. Analysis of GitHub in Software Development Processes

Using tools as GitHub[13] can have various effects on the general development process. Peterson (2013) analyzed the effect of GitHub on specifically the open-source development process. First, they wanted to know if in a GitHub project most of the contributions come only from a small subset of users. While their data largely supported this hypothesis, they also noted that there are still a lot of opportunities for other users to contribute to a project, even if it may not be in a major way. They also took a look at how GitHub is changing the open-source development process. They concluded that the most noteworthy change that can be attributed to GitHub is fostering a culture of knowledge sharing. Finally, they researched if GitHub is a viable

[13] github.com

platform for projects that have to adhere to certain standards. They showed that the tools that GitHub provides make it indeed possible for it to serve as a platform for high-quality open-source software projects.
Jarczyk, Gruszka, Jaroszewicz, Bukowski, and Wierzbicki (2014) examined 2000 GitHub repositories in terms of their quality. They used two measures to determine a project's quality. First, the attractiveness and popularity of a project, which is determined by how often a project is starred or endorsed by a user. The second measure is the quality of the support. This can be determined by looking at how long it takes the developers working on the problem to deal with project-related problems.
Having established these measures, Jarczyk et al. (2014) tried to find out what leads to a good GitHub project. In terms of popularity, they first looked into sheer activity but concluded that project activity is more likely to be an effect of a project's popularity. Projects that are owned by companies, however, seemed to be more popular on average than others. Another indicator of project popularity seems to be the number of followers the involved developers have. The higher the number of followers, the more attractive the project. The programming language has generally little effect on popularity, but not specifying a language has a negative effect. In terms of quality of support, the number of branches seems to a good indicator. The more branches a project has, the fewer the number of bugs. Also, the more repositories the developers own, the less likely they are to fix problems in a specific project.

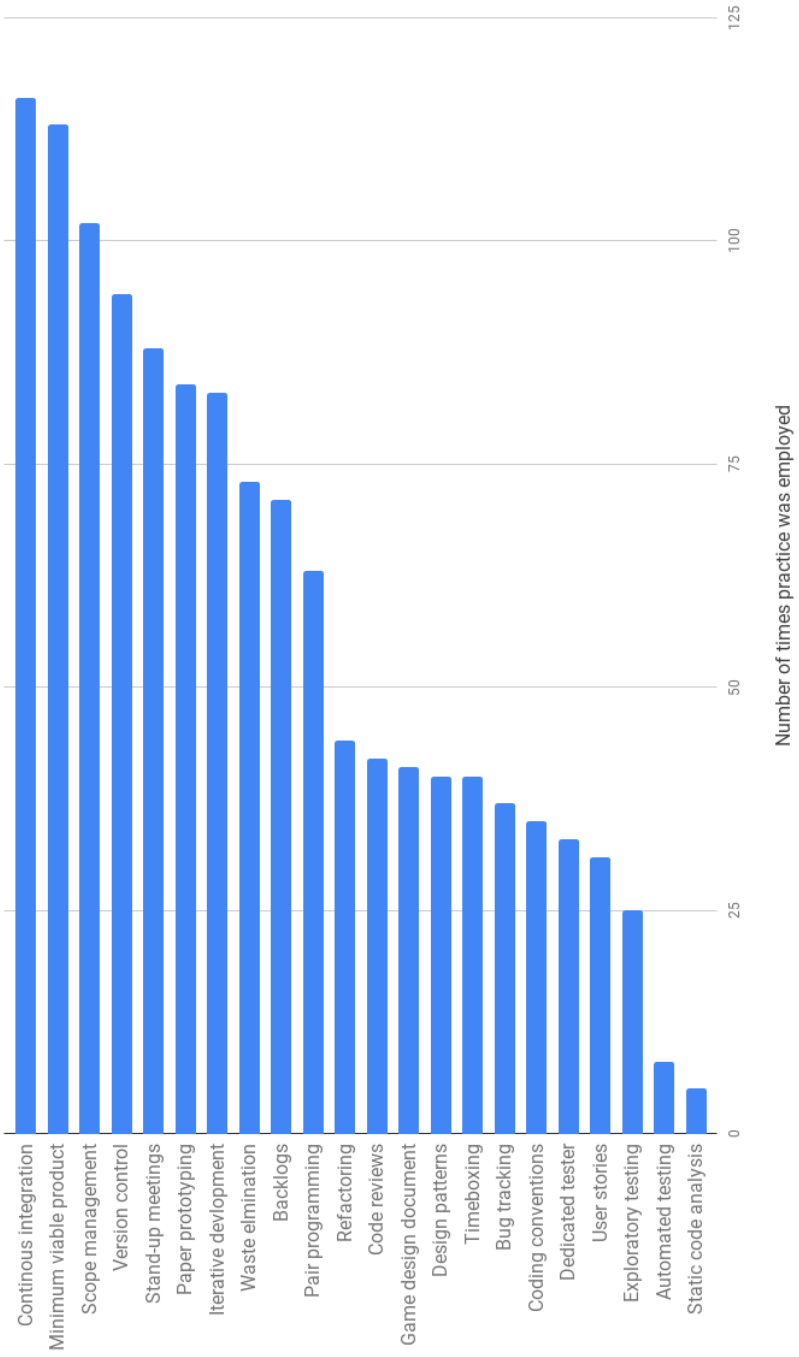

Figure 2.8.: Software development practices used by game jam participants, adapted from Borg, Garousi, Mahmoud, Olsson, and Stalberg (2019).

2.4. Related Work

Social networks can be constructed from a wide variety of sources, and different kinds of information can be derived from them. In this section, we will discuss some papers and take a look at how they constructed their networks and for what purpose.

Research on GitHub

Lima, Rossi, and Musolesi (2014) characterized large scale collaborations on GitHub. They analyzed more than 180 million events on public repositories over 18 months. They constructed a follower network of Github users as a directed network. Here the nodes represent the users and an edge symbolizes a follow-relationship between them. The network they created had over 600,000 nodes and over 200 million edges. This graph has an average degree of 3.019 and a clustering coefficient of 0.012, which as Lima et al. (2014) point out is lower than for example the average on Facebook or Twitter. This might have something do to with the fact that those social networks are commonly used for recreation purposes. GitHub, on the other hand, is a tool used for productivity. Lima et al. (2014) further took a look at the activity of the repositories. They found that only 62.90% of all scrutinized repositories had at least one commit on them. Further 74.22% of repositories have more than one committer, meaning that around a quarter of repositories is for one-man projects. In regards to the forks on repositories, they conclude that this feature is mainly used for a few popular key projects. Lima et al. (2014) also analyzed user activity and how it relates to their degree in the graph. They found that in general, a higher number of followers correlates with more activity. But they point also out that there are a lot of users that have many events associated with them without having a high follower count. The out-degree of the users has no bearing on their activity. Finally, Lima et al. (2014) analyzed the geographic locations of the users. Almost a third of the users in their dataset was from the USA, with the UK and Germany following in place two and three. They concluded that users are more likely to collaborate with other users from the same country, as a higher distance between them is associated with a higher cost. Generally, repositories with

a high number of collaborators center on a few key locations.

Research on Wikipedia

One of the most well-known collaboration platforms on the Internet is Wikipedia[14]. Laniado, Tasso, Volkovich, and Kaltenbrunner (2011) constructed three directed networks based on the English version of Wikipedia. The first network is the article-reply network, which consists of users as nodes and an edge is established between them if one user comments at least one entry by the other. The user-talk network is similar to the first, but an edge here symbolizes direct replies. The third network is the wall network, where an edge is formed between users by leaving a message on the personal talk page. The first two networks have very similar average degrees of 7.30 and 7.46 respectively. The average degree of the wall network is considerably lower at 2.37. Laniado et al. (2011) explain the low average degree of the wall network with users that write a lot of replies primarily direct them at Wikipedia newcomers.

Research on Corporate Email Traffic

Choosing a different approach than the two papers discussed in the two previous sections, Tyler, Wilkinson, and Huberman (2005) studied the peculiarities of an informal social network based on email traffic. The source of their data was their own company, HP Labs[15]. The network they constructed used the people as nodes and the emails they sent as the vertices connecting them. In total, their dataset consisted of 485 employees and 185,733 emails, which they used to construct a graph. Notably here is that when they constructed the graph they introduced a threshold of at least 30 emails being exchanged between two nodes to establish a link between them. The reason for this was to eliminate one-way relationships as much as possible. After

[14]https://www.wikipedia.org/
[15]https://www8.hp.com/us/en/hp-labs/index.html

this limitation, the resulting graph consisted of 367 nodes and 1,100 edges. One of Tyler et al. (2005) goals was to develop an algorithm that automatically detects communities in such structures. The idea behind this was to find informal work relationships, that may not be directly visible from the company's structure. Their method found 66 distinct communities in the network. When comparing that data to the main HP corporate directory, they discovered that 49 of these communities were entirely composed of employees of the same department. When looking at centrality measures of the individual nodes, it also appeared that nodes with higher centrality are associated with leadership roles. Tyler et al. (2005) further conducted interviews with sixteen randomly chosen employees that were part of the network and asked them if the communities their algorithm detected reflected reality. The feedback they received was largely positive here. The people that were part of cross-department communities also stated that the reason for the existence of those communities were either project or discussion groups. They also found that these communities included at least one manager. Believing their method to be an effective way to identify communities of practice, Tyler et al. (2005) hope to further enhance their algorithm by considering time stamps and using it in other contexts, like intelligence and covert networks.

2.5. Summary

Game jams are social game development events, where people from all kinds of technical backgrounds come together to create a game. Common rules in such events are a time limit for the development process and a theme for all developed games. Some popular events include the Global Game Jam, the Nordic Game Jam, and Ludum Dare. Game Jams pose a challenge for the teams in that they have to develop a functional game in a short time, usually 48 hours. The individual team members do not know each other beforehand in most cases, which is an additional hurdle.
Traditional software development models can not be applied to game jam events since they include long planning and requirement analysis phases before the start of the actual development. They are also concerned with future maintenance and continued development of the finished product.

In game jam events, there is simply no time for such development steps. However, some aspects are still applicable to game jams, like the multidisciplinarity of Scrum. Game jams create an environment that is unlike that of typical software development cycles. Everyone is encouraged to bring in their ideas, regardless of background. The risk of failure is also not a major concern at a game jam. Some other unique aspects of this form of development are the focus on product value, lightweight construction, and concurrent development. This exchange of ideas of people with unique backgrounds and experiences ideally leads to new innovative games. There is also a tangible benefit to the participants of the event. They provide an opportunity for students to put not only their acquired skills into practice but also the chance to learn completely new skills and tools. Further, game jam events are a good opportunity for networking and to meet new people from all kinds of backgrounds. In the next chapter, we will explain the methodology we used to get data about those people and their games.

3. Dataset and Preprocessing

As a particularly large game jam event with a well-maintained website, the Global Game Jam provides a good opportunity to learn more about jammers. In this chapter, we explain the origin of the data and how we acquired it for the analysis. We also discuss some of the limitations of the dataset.

3.1. Setup of the Global Game Jam Website

The information for jammers and games was collected from the official Global Game Jam page[1]. In the following, we will show what a profile looks like and what data we used from there. The crawl engine that we used for data acquisition is called Scrapy[2]. Scrapy works by giving the spider (the crawler program) a starting page, the information it is supposed to scrape from the site and how it should continue its crawl. The Global Game Jam website is organized in such a way that besides the main URL the sites are also numbered consecutively as nodes. So to start with, a complete site map (Table 3.1) of the Global Game Jame Site was produced, to find out which nodes correlate to the game profiles pages. The URL of game profiles is typically composed as https://globalgamejam.org/[Year of Game Jam]/games/[Title of Game]. The games of a particular year are ordered together, which makes it easy to determine the node range that needs to be crawled.

[1] https://globalgamejam.org
[2] https://doc.scrapy.org/

Short Link	Canonical Link
/node/1	/about
/node/2	/faq
/node/3	/games
...	...
/node/1000	/2014/games/island-chaos
/node/1001	/2014/games/murder-mansion

Table 3.1.: Short excerpt of the site map. The prefix of all URLs is https://globalgamejam.org.

3.2. The Profile Pages

This screenshot (Figure 3.1) shows the profile page of the game Xin[3] and serves as an example of what a typical game profile on the Global Game Jam website looks like.

The individual game profiles were the starting point for the data collection. From them, the following data was extracted:

G1. The **name of the game**, which is also contained in the URL of the profile.
G2. The **team members** that worked on the game, including the link to their personal profile.
G3. The **jam site** where the game was developed.
G4. The **year** when the game was made, also contained in the URL.
G5. The **tools and technologies** that were utilized during the project (optional).
G6. The **target platform** of the project (optional).
G7. The address of the **game's repository** (optional)

Additionally, the profile page may contain information such as screenshots and installation instructions. The following scrapy code snippet illustrates the crawling process in practice:

[3]https://globalgamejam.org/2017/games/xin

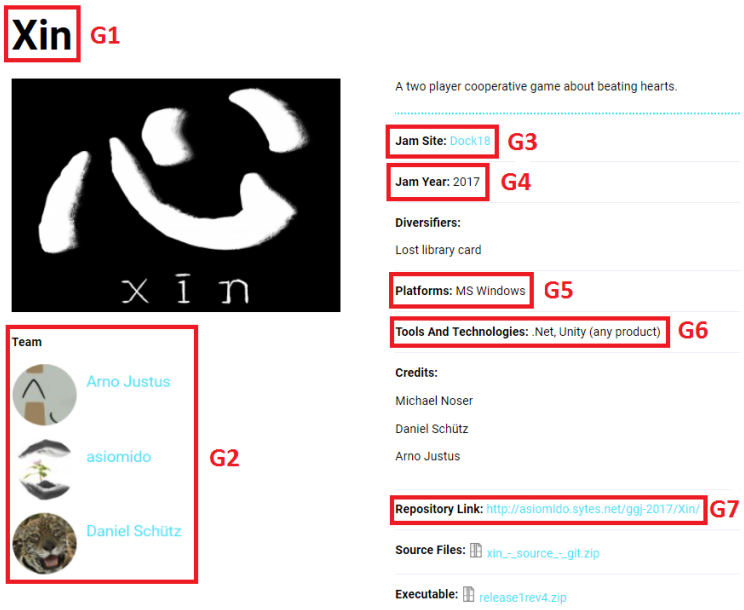

Figure 3.1.: Example of a Global Game Jam game profile page

```
def parse_games(self, response):
    def extract_with_css(query):
        return response.css(query).extract()
    yield {
        'title': response.xpath("//meta[@property='og:url']/
        @content").extract(),
        'node': response.xpath("//link[@rel='shortlink']/@href")
        .extract(),
        'team': extract_with_css('span.field-content a::attr
        (href)'),
    }
```

For every game, three properties were collected. The 'title' of the game, which is contained in it's URL, the shortlink 'node', which is only used for administrative reasons and finally the 'team' of jammers. Other attributes were collected on subsequent crawling processes. After the individual games, the profiles of the jammers that developed these games were scrapped.

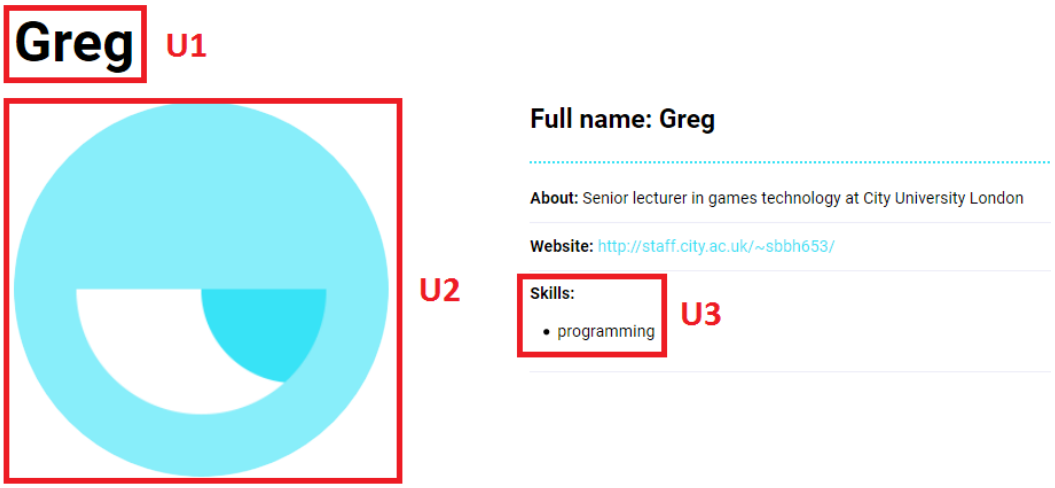

Figure 3.2.: Profile of the jammer greg

Figure 3.2 shows the profile of the jammer greg[4]. The main information that was valuable from the profiles were:

U1. The **name** of the jammer, which is also contained in the profile URL.
U2. A **profile picture**, used in the Team Recommender, Chapter 6.1 (optional).
U3. The **skills** each Jammer possesses. There are 15 different skills in total, and the jammers self-report if they are proficient in any of them.

Other information on the jammer profiles includes a short self-summary the jammer wrote, a link to a personal website, their full name and a list of previous Global Game Jam projects. Following that, the information of the individual Jam Sites was acquired (Figure 3.3). Of particular interest here were:

S1. The **name** of the jam site, also contained in the URL.
S2. A list of **jam site organizers and participants**.
S3. The **address** where the event took place.

An important step to generate the network was to create edge lists. Table 3.2 shows how the data is initially stored in the database, while Table 3.2

[4]https://globalgamejam.org/users/greg

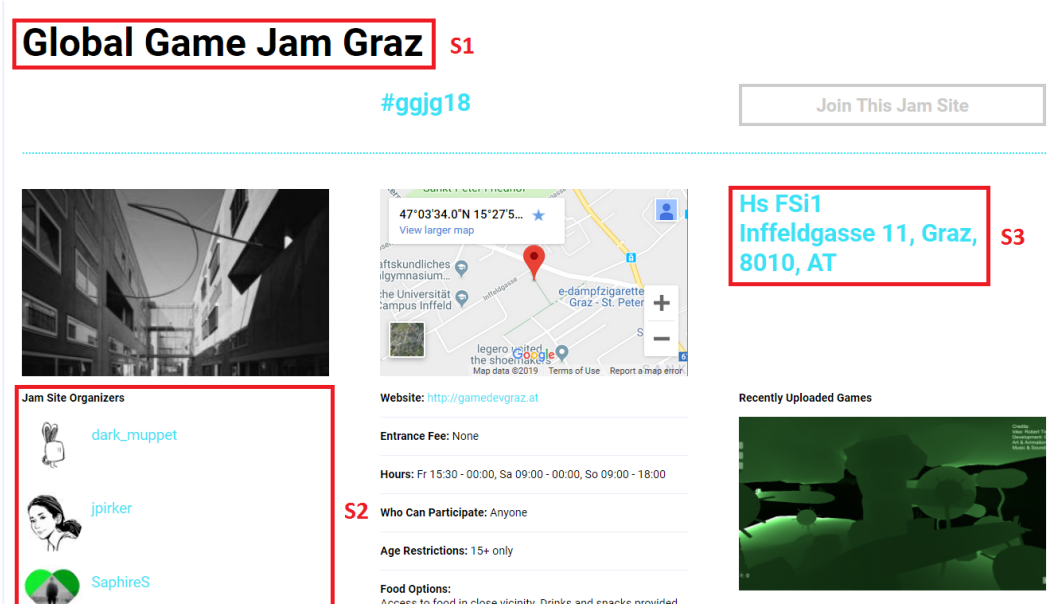

Figure 3.3.: Profile for the Jam Site of Graz in 2018

shows the resulting list. Next, for the GitHub data all of the collected game

Game	Developer
Game A	Jammer 1
Game A	Jammer 2
Game A	Jammer 3
Game B	Jammer 1
Game B	Jammer 3

Table 3.2.: Example of how the games and jammers are stored in the database.

repositories were scraped, with the exception of those that were not hosted on GitHub. However, some of the information gathered was not usable for scraping, for reason that are explained in more detail in Chapter 3.4.

Developer 1	Developer 2	Weight
Jammer 1	Jammer 2	1
Jammer 1	Jammer 3	2
Jammer 2	Jammer 3	1

Table 3.3.: Format of edge list for network creation.

3.3. Limitations of the Dataset

Even though the data was collected and analyzed with great care, there are still some limitations one has to consider when looking at the results. Since the information on the web pages was input manually, there can be discrepancies with reality. For example, the average degree for jammers from Trinidad and Tobago is 0, with only one game developed and only one person listed as the developer. However, when looking at the game's page, one can see a photograph of ten people, who also all have a profile page. These people probably are the actual development team, but since they have no game associated with them, they fail to show up in the dataset. Only one person, who happened to be the site organizer, is recognized as a developer. There are several more cases of this, however, for the sake of consistency these inaccuracies were left in the dataset. Furthermore, some games have no jammers listed under the team section at all[5]. If certain games omit some information, like for example the target platform, then those games will not be considered as a part of the analysis when that particular information is discussed. Also, there seem to be some games that are no longer listed on the Global Game Jam website. There are examples of games[6] that were part of the dataset that was discussed in Pirker, Lesjak, Punz, and Drachen, 2018, but this particular game is now seemingly removed from the website. In terms of skill analysis, it should be kept in mind that the skillsets of the jammers are based on their self-estimation. There is no way to quantify how good they are at a particular skill, or if it is even true that they have any expertise at all in the field in question.

[5]https://globalgamejam.org/2015/games/plan-b
[6]http://globalgamejam.org/2014/games/what-do-i-look

3.4. GitHub Data

In addition to the data from the Global Game Jam website, the data from repositories hosted on GitHub[7] was also scraped if the game page in question had a repository listed. However, not all of the repositories that were specified on the game pages could be used. In eleven cases, only the GitHub account name was listed, but not the specific repository that was used to host the Global Game Jam project. Meanwhile, the data from about 261 repositories could not be accessed at all. There are various reasons for this, like for example the repository being renamed or deleted altogether. Figure 3.4 shows an example repository[8]. Of primary interest were, of course, all the commits that were done to a repository, but there was also more additional data about the repositories collected, like the number of branches or the programming language used. Not all of this data was used for the data analysis, though. Further, as GitHub also has a follower/followee system similar to Twitter, this data was also scraped for each repository owner. The problem here though is that it is not possible to map the repository owner to a specific jammer that attended the Global Game Jam, since the repositories are only strictly connected to the game that is hosted on them. Nevertheless, a social network based on these connections was modeled for the sake of completeness.

3.5. Summary

We extracted the data for this paper from the official site of the Global Game Jam event. It contains information about the jammers, the games, and the jam sites since 2014. We started by crawling all the games using Scrapy. Of particular note here were the teams that developed them as the basis of the dataset and for the further construction of a network. Going from there, we collected the data of all jammers that were involved in the development of the known games. Here, the skills of the jammers were the primary interest for further analysis. Finally, we collected the data

[7] https://github.com
[8] https://github.com/dmitchell/ggj-war-of-gods

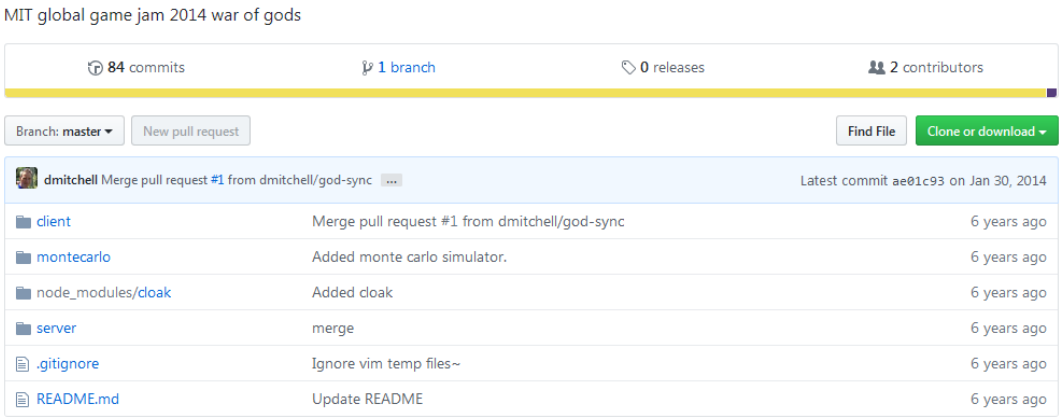

Figure 3.4.: The files and folders contained in a GitHub repository of a Global Game Jam project.

about the jam sites, including the jam site organizers and the address. We also briefly showed how we built the link table, which is the basis for the network. We then explained some of the limitations of the dataset, which primarily involves human input error. So in some ways, the dataset does not accurately depict reality. We also looked into if the games listed a repository link, and if that link points to a GitHub repository. We collected a multitude of data from the GitHub repositories, including the social relationships of the repository owner on GitHub, as well as the commits that were done on the repository that hosts their Global Game Jam project. The collected data will be used for various analyzes and constructing networks.

4. Network Structures and Characteristics

One of the uses for the dataset is to construct networks. In this chapter, we describe the characteristics of the networks by the collected data of the Global Game Jam website and GitHub and also discuss how they were built. The network of the Global Game Jam is weighted and undirected, while the GitHub network consists of unweighted, directed edges. To find out more about the jammers and their connections we will utilize parts of social network analysis.

4.1. Social Network Analysis

Prell (2012) explains that each person is part of multiple social networks, be it family, friends, or work colleagues. These networks used to be situated locally most of the time, but the rise of the Internet in the last 20 years has made it possible for people to build networks spanning all around the world. A social network can be represented as a graph, that consists of people and the connection they share between them. The focus of social network analysis then, as Otte and Rousseau (2002) state, is on the characteristics of these connections rather than those of the people themselves. Of course, the characteristics of the people are still important, since they can reveal more about the connections. Otte and Rousseau (2002) also characterize social network analysis as more of a broad strategy than an actual formal theory. Its major fields of influence are sociology, computer science, and mathematics. It is utilized in both physical and social sciences Borgatti, Mehra, Brass, and Labianca (2009) state, across various disciplines ranging from economics to psychology.

Some important concepts of social network analysis like centrality are explained below.

Network Concepts and Metrics

Here we will describe some basic network concepts and metrics, used by Gephi[1] and in the data analysis. All of the concepts explained in this section are according to citetDiestel2000 and Hanneman and Riddle (2005).

Graphs, Nodes, and Edges A network can be shown as a graph(G) where

$$G = (V, E).$$

Here, V stands for the nodes of the graph and E for the edges. In the case of social networks, the nodes often present the members of the network, while the edges indicate the connection between the members. In general, one can differentiate between two basic types of network, directed networks, and undirected networks. The difference here is how the edges connect the users. To give an example, a graph showing the friendships of a user of Facebook would be an undirected network. Similarly, the graphs in chapter 5 of the Global Game Jam relations are also an undirected network. On Twitter[2] on the other hand, the connections can be distinguished between followers and the person being followed. This results in a directed network. In directed networks, the direction of a node is typically denoted with an arrow. Furthermore, the edges in a network can be weighted or unweighted, In an unweighted network, the weight or the value of all edges is presumed equal. Counting the number of edges connected to a given node in an unweighted network determines the degree of a node. If a node has a degree of 0 means that it is not connected to any other node in the network at all. The average degree of a graph is the average of the degree of all its nodes and can be determined with

$$d(G) := \frac{1}{|V|} \sum_{v \in V} d(V)$$

[1] https://gephi.org/
[2] https://twitter.com/

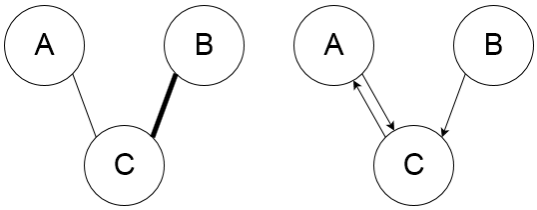

Figure 4.1.: An example of two similar, yet different networks. Both consist of three nodes and have a diameter of two. However, the graph on the left is an undirected, weighted network. The thicker edge between nodes B and C symbolizes a higher weight. The graph on the right is a directed, unweighted network. It should be noted that there is no connection from node C to node B here.

In the case of directed networks nodes have an in-degree and an out-degree. The in-degree states the number of edges leading to the node, while the out-degree gives the number of edges that originate from this node.

Network Diameter: The shortest path between to nodes in a network is known as their distance. If there is no connection at all between two nodes then their distance is set to infinity. The diameter is determined by the largest distance between two nodes in the network and can be used as a measure to how closely connected the network is.

Cliques: In a network with cliques, there are one or more subsets of nodes that share a closer connection to certain parts of the network than to others. A clique can be described as a maximum sub-graph of the network where every node is connected to one another. Naturally, the smallest number of nodes a clique can have is two, also called dyad. Commonly used are ”N-cliques”, where not all connections between nodes exist. Here, ”N” denotes the maximum distance between nodes.

Components: When looking at an undirected graph, a component of it is simply a subgraph where all nodes are connected, but not connected to other components. A subgraph is a subset of the nodes of the original graph, with all the edges that belong to them. There are two different types of components in directed graphs. Weakly connected components, where all nodes are connected, regardless of the direction of the edges. In strongly connected components, on the other hand, there must exist a direct path to each node in the component.

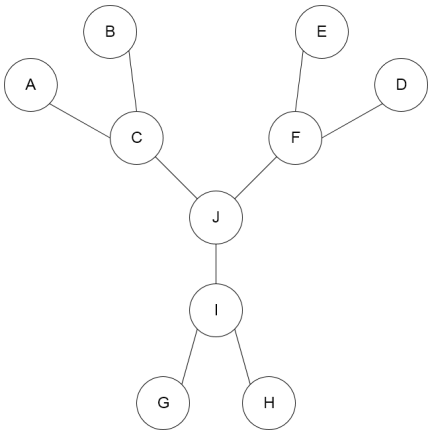

Figure 4.2.: The degree centrality of node J in this example graph is equal to that of the nodes C, F, and I. However, its position in the center of the graph grants it a higher closeness centrality as well as a higher betweenness centrality.

Centrality

Centrality is the term used for the influence a node has in a social network. There are several different ways to determine the centrality of a node.

Degree Centrality: This is the simplest way to determine the centrality of a node. It is equal to the number of edges that are connected to the node. In a directed network, one can additionally differentiate between an in-degree and an out-degree. Wasserman, Faust, et al. (1994) call a node with a high in-degree ”prestigious”, while having a high out-degree makes a node ”influential”.

Betweenness Centrality: Slightly more involved than the degree centrality, the betweenness centrality of a node is determined by how often the node shows up in the shortest path between all other nodes in the network.

Closeness Centrality: The closeness centrality tries to take into account the entire network to determine the centrality of a node instead of only the directly connected edges. The closeness centrality of a node can be determined by computing the average length between the node and all other nodes in the graph.

Eigenvector Centrality: The eigenvector centrality of a node can be determined by how well a node is connected to other nodes that have a high eigenvector centrality. The famous PageRank algorithm from Google is based on this centrality measure seen in Page, Brin, Motwani, and Winograd (1999).

Eccentricity: Eccentricity is the property of a node that is determined by its largest distance to another node. Thus the greatest possible eccentricity a node can have in a network is equal to the network's diameter.

4.2. Global Game Jam Jammer Network

Nodes	83,876
Avg. Degree	5.294
Avg. Weighted Degree	5.702
Edges	222,028
Diameter	65
Avg. Clustering Coefficient	0.883

Table 4.1.: Overview of the Global Game Jam jammer network (2014-2018)

Depending on how the data is mapped to the nodes and edges, different kinds of networks can be created. Pirker, Khosmood, and Gütl (2017) describe three different types of networks:

- In the **Jammer Network**, the jammers are mapped to the nodes, while the games they have worked on are the edges connecting them.
- For the **Location Network** the jam sites are connected thorough jammers by representing the sites as nodes and jammers as edges.
- The **Game Network** describes the jammers as edges connecting the games, which are the nodes in this case.

The Global Game Jam Jammer network is created from the data scraped of the Global Game Jam web page. In this network, as said above, the nodes represent the jammers that have attended a Global Game Jam. The edges that connect them represent the games that they have worked together

during the Global Game Jam event. Since this network has weighted edges, the number of games that the jammers have worked on together determines the weight of the edges. Jammers that have not worked with other people have of course no edges connecting them to the rest of the graph, but they still show up as unconnected nodes. In total, the network contains 83,876 nodes representing the people that developed at least one game at the Global Game Jam. The average degree is 5.294, which means that on average each jammer worked with around five other people during their time at the Global Game Jam. The average weighted degree of the jammers, which also takes into consideration how often they worked with the same people, is only slightly higher than their degree. We will discuss the implications of this in more detail in Chapter 5.1.1.
The network's diameter is very large at 65, but since the network is not fully connected there is no connection between all jammers. However, the average clustering coefficient of the graph is 0.883. An average clustering coefficient of 0 means that no nodes are connected at all, while 1 means that all nodes are connected. So the average clustering coefficient still points to a well-connected network. In comparison to the analysis done by Pirker et al. (2018) that included Global Game Jam data up to 2017, the average degree and average weighted degree are higher, and at the same time the diameter has decreased from 73 to 65.
Taking a look at the degree distribution in Figure 4.3 it should be noted that 91.28% of the jammers have a degree of ten or lower and 64.55% have a degree of five or lower. Regarding the weight distribution, it can be noted that 93.79% of the edges in the network have only a weight of one. The maximum edge weight in the network is nine, and there is only one of it in the graph. Notably, there is no edge with a weight of eight at all.

4.3. GitHub Network

The setup of the GitHub network is quite different from the Global Game network. This network is built from users following each other on GitHub, so the nodes represent the individual GitHub users. The nodes in this network are directed though, the direction of the edge indicating who is

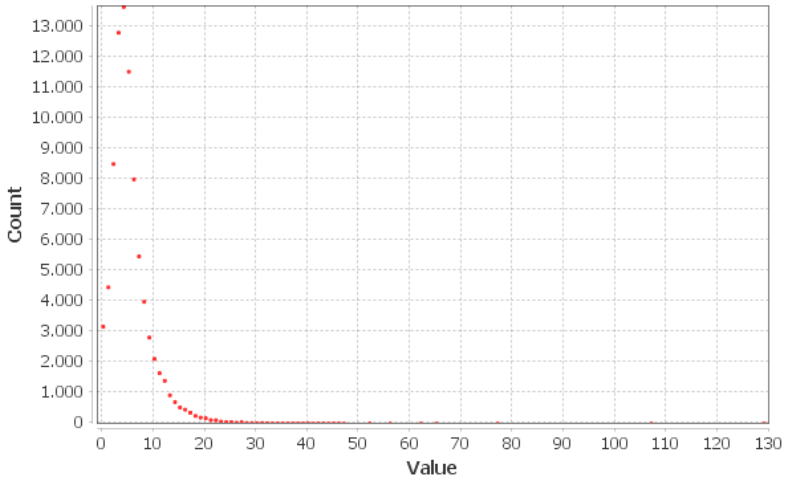

Figure 4.3.: Degree Distribution in the Global Game Jam network

following whom. Since it is not possible to follow a user multiple times the edges are of course unweighted.

Nodes	35,982
Avg. Degree	1.341
Edges	48,269
Diameter	25
Avg. Clustering Coefficient	0.018

Table 4.2.: Overview of the GitHub network.

Consisting of 35,982 nodes in total, the first thing that stands out about the GitHub network (Table 4.3) is that it is a relatively sparse network. The average degree is only 1.341, meaning that a user only follows or is followed by slightly more than one other user on average. The average clustering coefficient is low as well at 0.018, being close to zero. The starting point of this network was the 2892 jammers that hosted a project during the Global Game Jam. That means that 91.96% of the nodes in this network are ”external” users, meaning accounts that have no direct association with the Global Game Jam event. Excluding them, the average degree of the jammers is 23.34 (68.77). Since this is a directed network, the in- and out-degree can

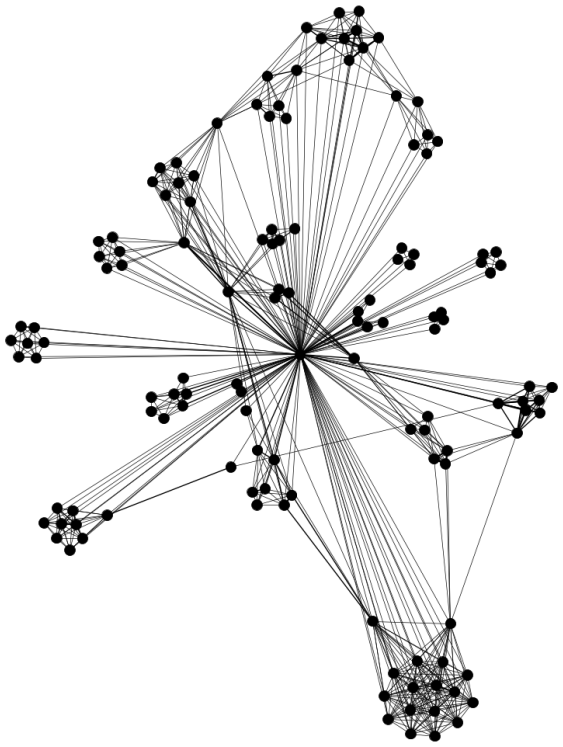

Figure 4.4.: Excerpt from the Global Game Jam network. Here all the neighbors of the jammer with the highest degree in the network are shown.

also be compared. On average, jammers are followed by 9.58 (24.84) other users, while following 13.76 (57.98) users themselves. Even though the full network has nearly 50,000 edges, only 587 (or 1.22%) of them are actually between jammers that used a GitHub repository. Therefore, it is probably safe to conclude that participating together in a Global Game Jam has only a low probability to lead to a connection on GitHub.

4.4. Summary

Social network analysis is an approach to discover various qualities of the connections people form. In this chapter, we described some basic network attributes and metrics related to the data analysis. The most important metric for this work is the measurement of centrality. Centrality refers to the amount of influence a node has in the network. The type of centrality measurement we will most commonly use in this work is the degree centrality. In an undirected network, this is simply the number of edges a node possesses. In a directed network, one can further differentiate between the in-degree and the out-degree.
We also explained the Global Game Jam and network and the GitHub network, their general attributes and how they differ from one another. The Global Game Jam network is an undirected, weighted network. It represents the jammers that worked on at least one game as nodes. If two jammers worked on a game together, an edge is established between them to show their connection. The number of times they worked together determines the weight of the edges in this graph. Even though the diameter of the Global Game Jam network is large, its clustering coefficient still points to a well-connected network. The GitHub network, on the other hand, is only very sparsely connected. The nodes in this directed, unweighted network are the individual GitHub profiles of jammers, while the edges show who they are following or are being followed by. Also, even though the repositories of jammers were the starting point of the network, there are only very few connections between them in this graph. There is a lot of other information that one can extract from these networks.

Figure 4.5.: Excerpt from the the GitHub network. This is the ego network of the node with the highest in-degree.

5. Data Analysis and Results

After having collected the game jam data and constructing a network from it, in this chapter we explore them to determine various characteristics of the jammers that attend the Global Game Jam and their usage of GitHub to answer the research questions.

5.1. General Data

Year	2018	2017	2016	2015	2014
Countries	102	89	89	75	67
Sites	767	669	616	503	460
Jammers	28,997	25,700	24,618	19,863	16,052
Games	8,575	7,192	6,856	5,430	4,203

Table 5.1.: Overview of the Dataset

First, an overview of how the Global Game Jam event evolved in terms of numbers. Table 5.1 describes how many countries, sites, jammers, and games are represented at each year of the Global Game Jam. Since this data comes from crawling the site, it may differ from official data that was published elsewhere. As we can see, the Global Game Jam is in a state of steady growth, with more countries and people participating, and more games being developed.

Skill Distribution

First, a look at the general skill distribution. The 83,876 jammers have on average 3.35 skills, with 138 (0.16%) jammers listing all of the 15 skills

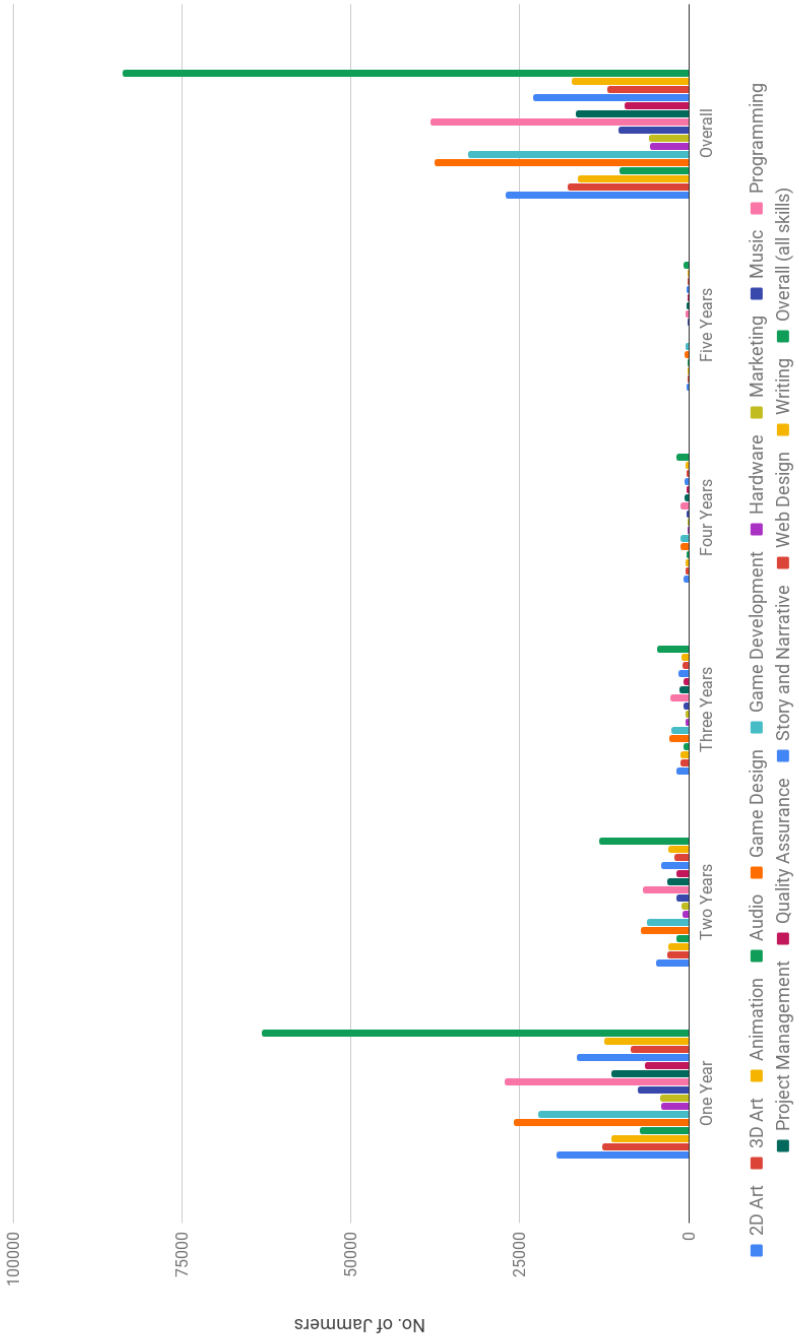

Figure 5.1.: Number of jammers with a certain skill and how many Global Game Jams they have visited. Jammers with programming skills are very common.

on their page, whereas 14,823 (17.67%) jammers do not state any skills at all. From the 15 skills, Programming is the most common one, with 38,268 (45,62%) jammers describing themselves as being proficient in it, nearly half of all them. Programming is closely followed by Game Design (37,632 jammers, 44.87%) and Game Development (32,702 jammers, 38.99%). The least common skills among jammers are Hardware and Marketing, being possessed by only 5,697 jammers (6.79%) / 5,845 jammers (6.97%), respectively. Programming is also the most common skill among jammers that have only attended one Global Game Jam, whereas for jammers that have been participating in Global Game Jams for five years it only ranks second with 499 jammers, being overtaken by Game Design with 538. In general, the average jammer visited 1.3764 (0.7716) Global Game Jams. The vast majority of 63,171 jammers only participated in one Global Game Jam, with 20,705 jammers (24.69%) attending two or more Global Game Jams.

Skill and Degree Growth

Jammers that only visited one Global Game Jam have on average 3.13 skills, while those jammers that took part in two Global Game Jams have 3.81 skills and those with three Global Game Jams under their belt 4.29 skills.
So far, the more Global Game Jams a jammer attended, the more skills do they possess, but the group of jammers that were involved in four Global Game Jams in total has with 4.59 on average slightly more skills than those jammers that participated in all five analyzed Global Game Jams at 4.53 skills.

Figure 5.2 shows the average degree of the participants regarding their skills and their attendance. Regardless of the skill set, the degree of the participants increases the more Global Game Jams they attend. The two skills with the lowest average degree are Programming (5.0011) and Web Design (5.0308), while Audio (5.7863) and Music (5.7052) have the highest. Possible interpretations for this are that skills like Programming are fundamental to development, that even in one-person-teams have to be represented. Audio and Music are comparatively more auxiliary and make it possible for the participants with those skills to work at multiple projects at once, thus increasing their degree. Another possibility is brought up by Pirker and Voll,

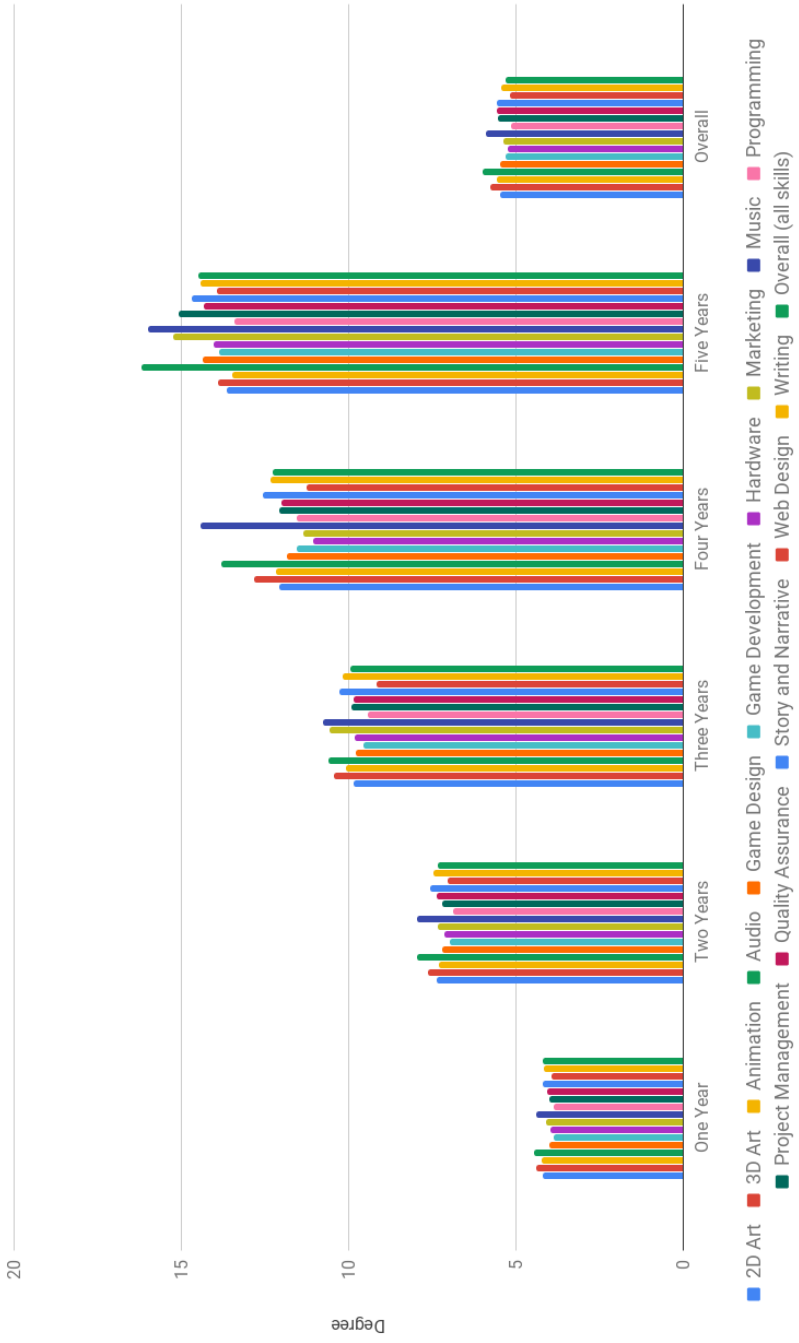

Figure 5.2.: The development of jammer's degree over the years, separated by skill. As can be seen, the degree jammers with audio and music skills grows slightly faster.

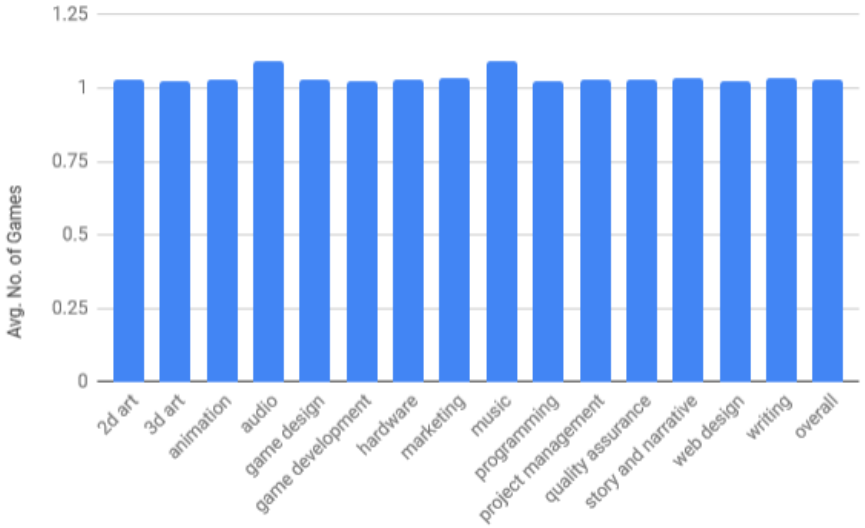

Figure 5.3.: This graph shows the average number of games a jammer with a particular skill makes during a Global Game Jam. Audio and Music both stand out.

2015. They observed that at the Vancouver Global Game Jam event jammers who are experienced in the field of Audio or Music were in short supply. This resulted in those jammers having to *"run around from team to team"* to help out with the audio. In any case, when looking at the average number of games per Global Game Jam a jammer develops, those who have skills in Audio and Music have indeed a higher number of developed games than all other skills. Most skills have an average of about 1.02-1.03 games developed per Global Game Jam, while jammers with skills in Audio and Music have an average of around 1.09 games (Figure 5.3).

We have shown that having attended multiple Global Game Jams correlates with a higher degree and also with a higher number of skills on average. But does a high number of skills alone also lead to a higher degree in the network? As can be seen in Figure 5.4, this is not the case. There is no correlation whatsoever between the number of skills a jammer possesses and their degree (ρ = 0.0223). The jammer with the highest degree in the network[1] only has two skills, but these two skills are Music and Audio,

[1] https://globalgamejam.org/users/grag

which are both associated with a high degree. This jammer also participated in four of the five assessed Global Game Jams.

We also looked into if country indicators such as population or GDP per capita influence the average degree of the jammers of the country. The population of a country seems to not affect either way on the average degree of its jammers, with a Pearson correlation of -0.0211. GPD per capita shows only a slightly higher correlation of 0.1736. Even when using Global Game Jam specific data, like the number of jammers per country there is no sign of a strong correlation, the Pearson correlation, in this case, being 0.2698. The most probable cause for a specific country having a high degree is rather obvious: the number of Global Game Jame events that took place there. Here, the Pearson correlation shows up as 0.5177. The tables containing the individual average degree per country can be found in Appendix Chapter A.

Which attributes of jammers lead to a higher degree? One factor that can predict the degree of a jammer is the type of skills they possess. As mentioned, particularly skills related to Audio and Music are associated with a higher degree. The pure amount of skills, however, is largely irrelevant. A second factor that also contributes to a higher degree is the number of attended Global Game Jams.

5.1.1. Degree Development over the Years

While the degree of a jammer represents how many different people they have worked with together, the weighted degree also takes into account if jammers collaborated multiple times, which in most cases means across multiple Global Game Jam events. The overall average weighted degree of 5.7024 is barely higher than the overall average degree of 5.2945. This can be explained by the fact that again most jammers only have attended one Global Game Jam, so there is no chance to collaborate with the same people again. In terms of skill sets, there are no real surprises either, with skills with a high degree also tending to have a high weighted degree. What is more interesting to look at is how the weighted degree changes over time and how this development compares with that of the degree, as seen in

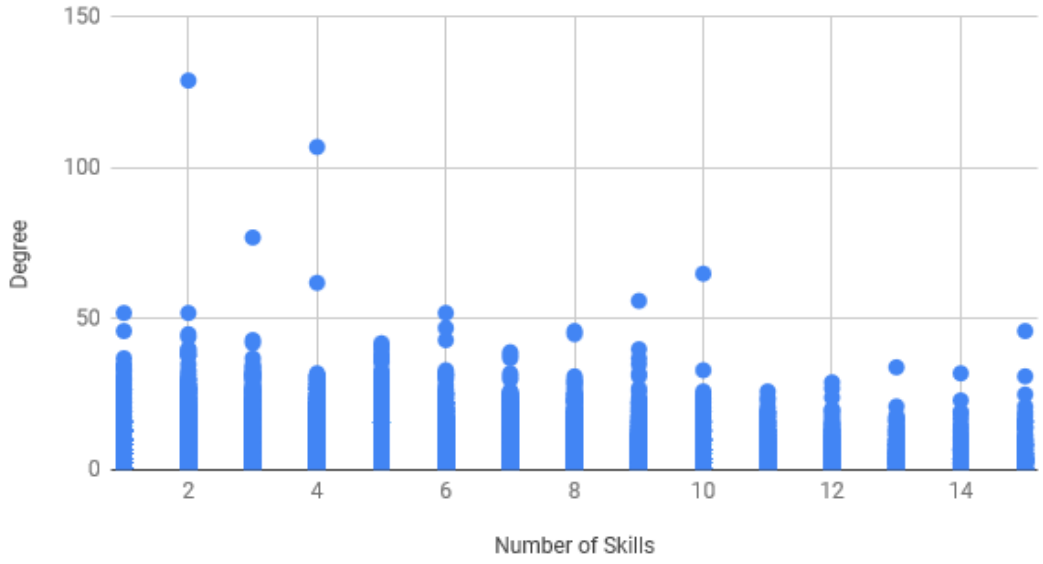

Figure 5.4.: In this scatter plot the correlation, or better the lack of a correlation between the number of skills a jammer has and their degree is displayed.

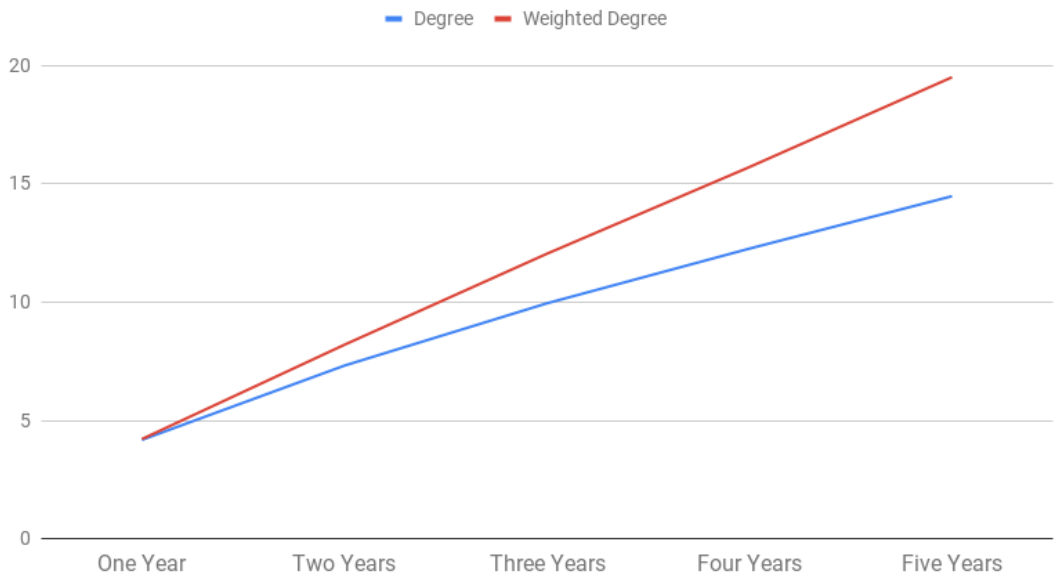

Figure 5.5.: Comparison of degree growth against weighted degree growth. The weighted degree grows slightly faster than the degree.

Figure 5.5. On average, the degree of jammers increases by roughly three for each Global Game Jam they attend. Their weighted degree, however, increases by about four for each year they participate in a Global Game Jam. Since the degree only increases when new connections are formed, it can be concluded that jammers tend to develop games with people that they have not worked with before instead of sticking with the same group of people over the years.

Do jammers who attended multiple Global Game Jams stick with the same team members? Based on the development of the degree to the development of the weighted degree, jammers do seem to get to primarily work with new team members at each Global Game Jam. A large factor for this could be the low average number of Global Game Jams (1.3764) jammers attend, which would make forming new teams a necessity.

5.1.2. Technologies and Platforms

The choice of utilized engines and tools, as well as the target platforms are fundamental decisions jammers have to make at the start of their project. Here, we will discuss what choices they have made.

Technologies

Year	Number of Technologies	Average per Game
2014	9	1.0057 (0.0751)
2015	14	1.0262 (0.1598)
2016	22	1.0668 (0.2968)
2017	21	1.0899 (0.2968)
2018	34	1.0682 (0.2709)
Overall	35	1.0595 (0.2498)

Table 5.2.: Utilized Technologies for Ggames

The number of deployed technologies grew swiftly over the years, starting with a mere nine in 2014 and reaching 34 in 2018. Overall, 35 different technologies have been deployed by jammers, meaning that in 2018 every single technology that was used to develop a game in previous years was also used in 2018 with the sole exception of Enchant.JS. Enchant.JS was one of the nine technologies used in 2014 but saw a steady decline over the years. Some technologies, however, are very niche, for example PlayStation Mobile is only used by one game in 2018 after not being used at all in 2017. Six technologies have been used throughout every single Global Game Jam. Those are Unity, GameMaker, Unreal Engine, Game Salad and Construct. It is very unusual for a game to utilize more than one technology, there are only about 60 that use more than two.

Unity is by far the technology that is most often used by jammers for their Global Game Jam projects. It also sports a relatively high degree of 7.0606, compared to 5.294 for the overall network. As shown in Figure 5.6, the used technologies cover quite the range of degrees, the highest being Houdini with 10.5025.

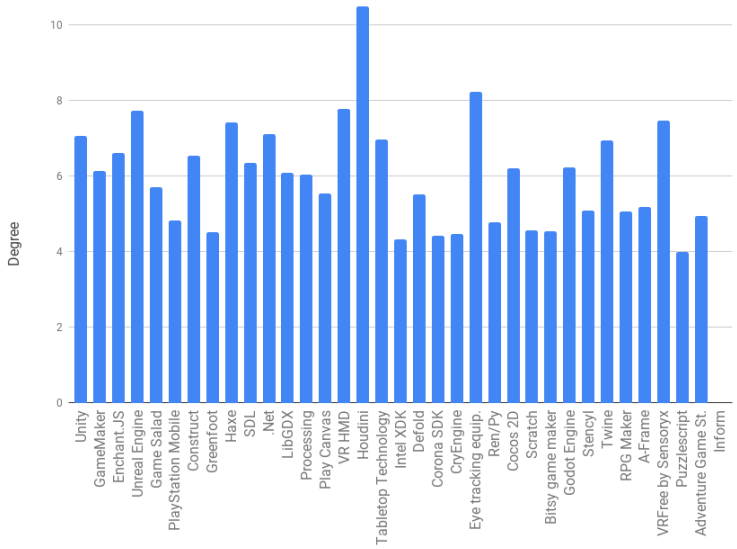

Figure 5.6.: Average degree of technologies. The technologies on the left are more commonly used. The differences in degree here are much more pronounced than in the equvivalent platform comparison.

Platforms

Year	Distinct Platforms	Average per Game
2014	17	1.7964 (1.2979)
2015	22	1.8894 (1.5067)
2016	22	1.6920 (1.2976)
2017	22	1.5871 (1.1851)
2018	26	1.4998 (1.1012)
Overall	26	1.6646, (1.2708)

Table 5.3.: Target Platforms for Games

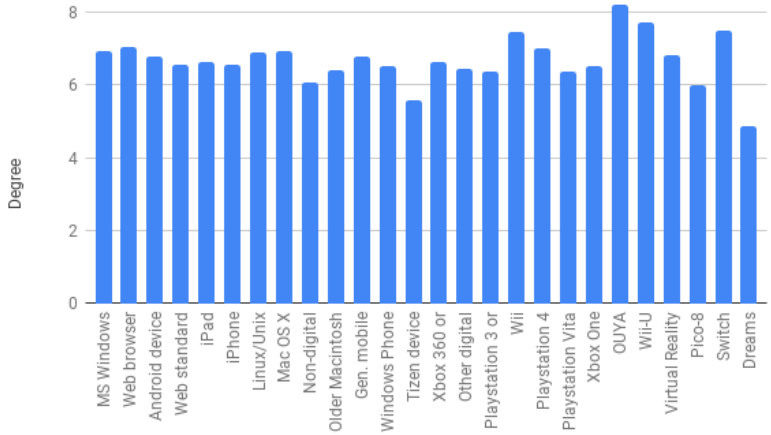

Figure 5.7.: Average degree of platforms. The platforms on the left are more commonly used. There is not much of a difference in degree among the most popular platforms.

When compared to the utilized technologies, the target platforms the games were developed for remained relatively stable. Also, in 2018 there were not any dropped platforms at all, each of the 26 platforms that have been developed for in previous years also received a game in the latest Global Game Jam. The number of targeted platforms per game, however, declined through time, even though only very slightly.

MS Windows and Web Browsers with special plugins or packaged apps are the most popular platforms to develop for during a Global Game Jam. Opposed to technologies the targeted platform does not seem to have much influence on the degree of the involved jammers. The outliers, like OUYA on the higher end and Dreams by Media Molecule on the lower end, are seldom developed for.

Details on Technologies and Platforms

For the following, only games that listed both the technology that was used and the platforms the games were developed for are taken into account. This applies to 24,719 games in total. Regardless of technology, a game is targeted

for 1.7802 platforms on average, which is remarkably close to the average of Unity of 1.7085. Notable, but perhaps not surprising, seeing as Unity is again by far the most popular technology, being used by almost 80% of all games considered here. Also, 87.57% of all Unity projects have MS Windows as one of their target platforms. Indeed, there is barely a country in the world where Unity is not the most used technology and/or MS Windows the most popular target platform. The only countries where MS Windows is not the top platform are Cambodia, Cameroon, El Salvador, Jordan, Trinidad and Tobago, and Zambia. Unity is not the most used technology in Algeria, Cameroon, Singapore, and Zambia. All these countries have in common that the total number of games is rather low, with only Jordan (115) and Singapore (256) having more than 100 games. Cambodia, Cameroon, El Salvador, and Trinidad and Tobago also all only participated in one Global Game Jam in total.

Which technologies and platforms are used at the Global Game Jams? As shown, the used platforms and technologies that are utilized by jammers at the Global Game Jam cover a wide variety, including some really niche products. However, the use of the Unity engine with MS Windows as the target platform is prevalent in almost every participating country. Global Game Jams are still an opportunity for fans of rarely used or obscure technologies and/or platforms to develop a project for them.

5.1.3. Teams of One

Even though the projects at the Global Game Jame are typically developed in teams, some people may prefer for various reasons to work on them on their own. Some jammers never collaborated with other people even though they might have participated in multiple events. In other words, these are the jammers with a degree of zero. In total, 3,179 jammers fall into this category. Unsurprisingly, most of these jammers have only participated in one Global Game Jame with an average of 1.1003 (0.3725), lower than the global average of 1.3764 (0.7716). In detail, there are 257 jammers with a degree of zero who took part in more than one Global Game Jam. There is only one jammer in the dataset that participated in all five analyzed Global Game Jams and still

possesses a degree of zero[2]. The lone developers produced 3551 games, of which the jammers with multiple attendances are responsible for 579. In terms of GitHub usage, they utilized 221 repositories.

Now, let us take a look at how the skills of these lone wolves compare with the more team-oriented jammers. Jammers with a degree of zero have 4.7354 (2.8954) skills, which is higher than the 4.0677 (2.4186) of the other jammers. The chart in Figure 5.8 lists in more detail which skills they possess. When comparing how common a skill of the zero-degree group with the other jammers is, the ranking is almost identical. The only exceptions are that Music and Audio are switched and people that are skilled with 3D Art more commonly work in teams. However, when looking at the relative share of jammers possessing a certain skill, the zero-degree groups scores higher in most cases. This is especially notable in the top three common skills, Programming, Game Design, and Game Development. By contrast, the general group of jammers has a higher ratio of people skilled with 2D Art, Story and Narrative, 3D Art, and Animation. The conclusion here is that jammers in the zero-degree group commonly have the core skills necessary to develop a game, but lack skills that belong in more artistic areas.

In terms of technology, jammers with a degree of zero utilized 33 of the 35 technologies that were overall used to develop projects in the Global Game Jams events. While in absolute numbers Unity is the most used technology again, a closer look at the data uncovers some interesting details. The technology Inform is only used by one game, which was developed by a single developer who has a degree of zero. Similarly, the Bitsy game maker was used to develop eleven games overall, and of those ten were made by single developers of which seven were developed by jammers with a degree of zero. The only other game that used the Bitsy game maker was made by a team consisting of only two jammers. Curiously, jammers with a degree of zero generally seem to be more likely to state the technology they used for their projects on the respective website. Jammers with a degree of zero make up 3.79% of all jammers, but even the technology with the lowest ratio of projects developed by jammers with a degree of zero (Eye tracking equipment) has a share of 5.26%. The same applies to the platforms they target. Jammers with a degree of zero developed games for all of

[2]https://globalgamejam.org/users/lightguard

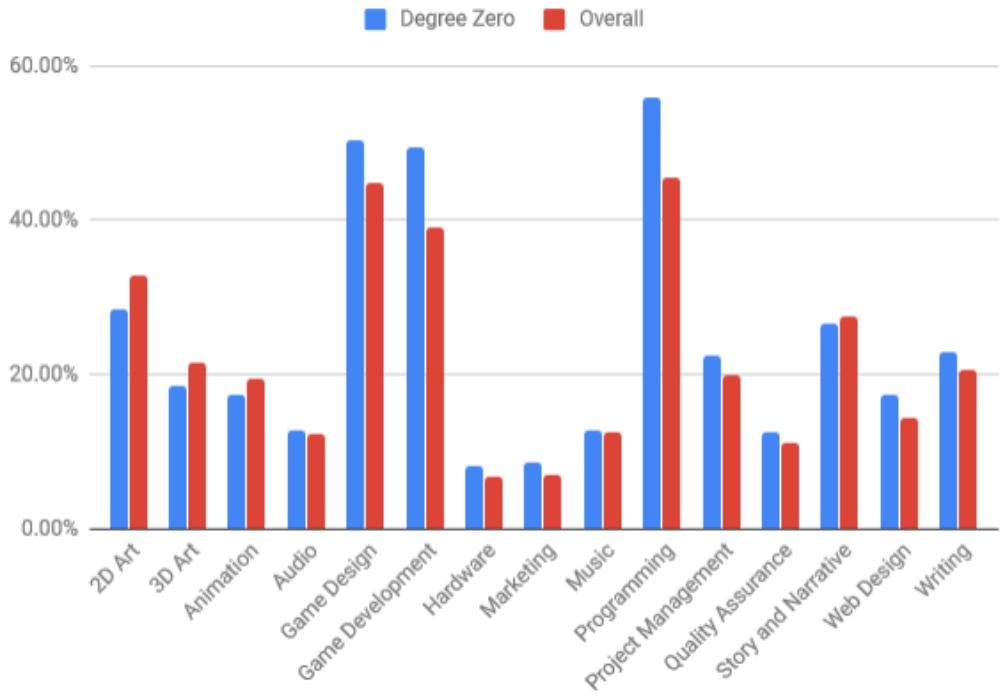

Figure 5.8.: Comparison of skill distribution between jammer with a degree of zero and the overall jammers.

the 26 target platforms of the Global Game Jam events. There is again no difference in absolute numbers between jammers with a degree of zero and other jammers, with MS Windows being the most popular platform overall. Relatively speaking, Dreams by Media Molecule is a notable platform as of the only five games that were developed for this platform, two were made by jammers with a degree of zero, making it the platform with the highest share of games developed with a degree of zero. Further, console platforms like the Xbox or the Playstation appear to be popular with jammers with a degree of zero.

How do jammers with a degree of zero differ from other jammers? In general, jammers that do not work in teams are a rarity, but the few that do attend Global Game Jams tend to have more skills than other jammers. Core development skills like Programming are also more common in this group. Working on their own also allows them to utilize a wide and in some cases

obscure range of platforms and technologies. However, they are even less likely to attend multiple Global Game Jams than other jammers.

5.2. GitHub Usage

In this section, we take a look at how jammers make use of GitHub[3] during the game jam event and beyond.

5.2.1. About GitHub

In today's world of software development, some form of version control and web hosting of source code is almost a necessity for projects that have multiple team members. But even single developers can benefit from the features such systems provide. One particularly popular system is GitHub. The company GitHub was founded in 2007 and in June of 2018 the company was eventually acquired by Microsoft. On GitHub, users can create repositories to upload their code so multiple people can work on the same project. GitHub can also be used to distribute the project to a wider public.

In the following, only the usage of GitHub repositories will be considered. However, there are a few other services that jammers use to host their projects. In total, 4,464 games have a repository, of which 3,680 (82.44%) are hosted on GitHub, while the remaining 784 repositories (17.56%) are located on quite a diverse selection of other services, the most popular being bitbucket.org with 404 repositories. As can be seen in Figure 5.9, the number of GitHub repositories grew rapidly over the years and much faster than the total amount of repositories hosted on other providers, which dropped off a bit from 2017 to 2018. GitHub is therefore by far the most popular service for jammers to host their projects. Of the 4,464 games with a GitHub repository, 11 game pages only listed an account name and no specific repository, while another 261 repositories were either deleted or renamed. These repositories will naturally not be factored into the statistics. For GitHub users, Unity is

[3]https://github.com/

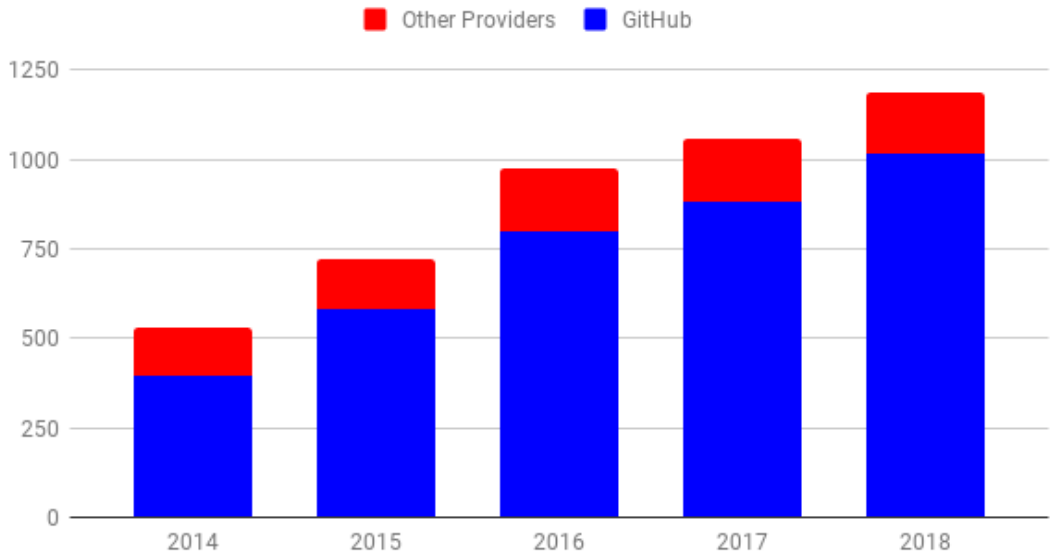

Figure 5.9.: Comparison of repository providers. Not only are most repositories hosted on GitHub, but the number of repositories each year is growing faster than those of other providers as well.

again the most popular technology, while the selection of technologies is smaller at 23. Platforms are more diverse with every single platform that is featured in the Global Game Jam also appearing on GitHub, including non-digital games, but MS Windows is as expected also the most common target platform here.

5.2.2. Commit Behaviour

Year	Repositories	Average Commits
2014	398	92.0377 (157.6267)
2015	580	92.1415 (143.0082)
2016	801	88.3610 (121.7298)
2017	883	75.6557 (73.2696)
2018	1018	83.0479 (114.0057)

Table 5.4.: Average Number of Commits

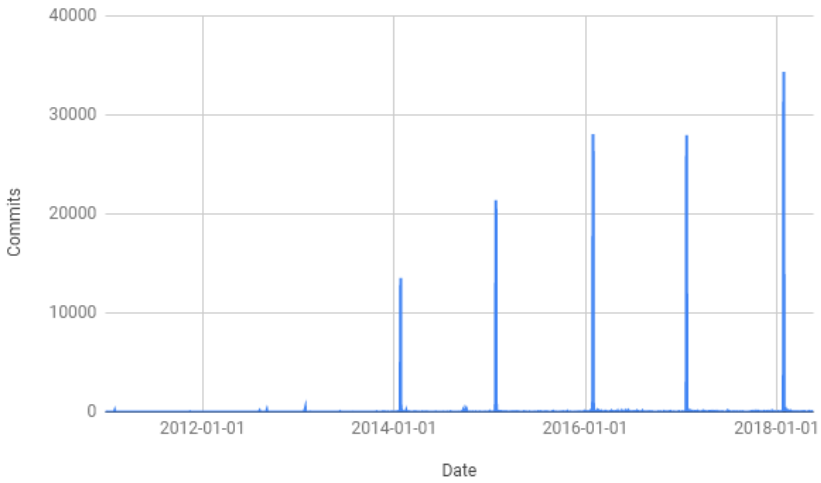

Figure 5.10.: Timeline of commits to Global Game Jam repositories. Each spike in the graph takes happens during a Global Game Jam event. Aside from that, there is barely any activity.

The number of repositories used is steadily increasing but as shown in Table 5.2.2 there does not seem to be any pattern to the average number of commits a team produces, ranging from 75 to 92. The repositories used by the previously discussed jammers with a degree of zero, however, sport a drastically lower average commit count at 31.6859.

Year	Repositories	Total Commits	During Global Game Jam
2014	398	31753	24608 (77,50%)
2015	580	48190	40571 (84,19%)
2016	801	62913	54010 (85,85%)
2017	883	64338	56300 (83,31%)
2018	1018	73840	68443 (87,51%)
Overall	3680	281034	243932 (86,80%)

Table 5.5.: Commits during the Global Game Jam

On average, a team that hosts its project on GitHub consists of 4.1162 (2.1701) members, while the average number of unique users that commit to the repository is 3.493 (2.5807). So it can be concluded that almost all of the jammers that work on a game also commit changes to the respective

Year	Total Repos	Six Months	One Year
2014	398	38	31
2015	580	53	41
2016	801	58	37
2017	883	56	29
Overall	2662	205	138

Table 5.6.: Repositories that received Commits six months or later after their Global Game Jam

repository. Most of the commits to the repositories are indeed made during the three days of the Global Game Jam itself and the percentage has been relatively stable since 2015 (Table 5.2.2). The diagram in Figure 5.10 clearly shows that outside of the days of the Global Game Jam event there is barely any activity on the repositories. Even so, some teams still work on their projects after the Global Game Jam has concluded. After one year, the vast majority of games are no longer worked on and do not receive any updates anymore, pending the possibility that someone is still working on them locally or on a different repository (Table 5.2.2).

5.2.3. Characteristics of GitHub Users

The average degree of jammers that utilize GitHub of 7.167 (5.873) is higher than the global average of 5.295 (3.938). Their weighted degree of 8.139 (7.031) is higher as well. This can be taken as an indicator that jammers that host their projects on GitHub have a deeper engagement with the event than those that do not. In terms of skills, GitHub users have on average 5.0474 skills, even more than the jammers with a degree of zero. Further, Programming is significantly more common with GitHub users than with other jammers, even though Programming is already the most common skill among all jammers. Interestingly, the skills Audio and Music which are associated with a high degree are also slightly more common among GitHub users. Jammers that are associated with a GitHub project have attended significantly more Global Game Jams on average than those that are not, that average being 1.9239 (1.1594) Global Game Jams. This is another indicator that GitHub users are more engaged with the event than

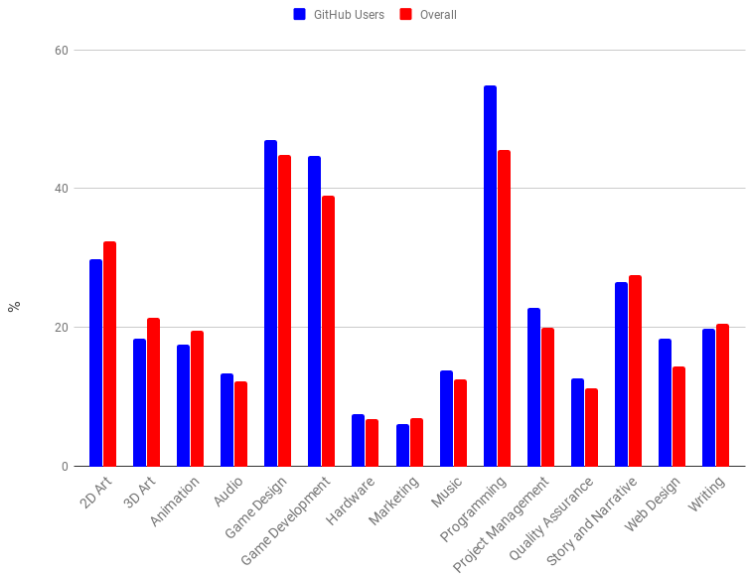

Figure 5.11.: Comparison of skill distribution between jammer that use GitHub and the overall jammers.

others. The repositories have some additional data associated with them that were not available on the Global Game Jam game pages. So we can, for example, take a look at the used programming languages. In total, 51 different programming or scripting languages were used in projects. C# is the most popular language being used in 1,802 projects, more than half of them.

5.2.4. Global Differences in GitHub Usage

Finding patterns for what a specific country makes it more likely for its jammers to use GitHub is tricky. Neither the average degree or weighted degree of the jammers, nor the number of Global Game Jam events that were held in a country seems to strongly correlate with the number of GitHub repositories. More developed games, of course, lead to more GitHub repositories being used, but the number of games hardly influences the relative number of projects that are hosted on GitHub. Nevertheless, GitHub is rather commonly used at the Global Game Jam events, with 88 of the

overall 107 countries having at least one GitHub repository. The United States, the country with the most jammers and the highest amount of developed games naturally also sports the most GitHub repositories. 1,066 GitHub repositories originate there, which is 29.08% of all Global Game Jam GitHub repositories. The countries with the next most repositories, Canada and the United Kingdom are already trailing relatively far behind at 272 and 252 repositories respectively. However, these are still impressive numbers seeing as the per-country average is only 34.33 repositories. When it comes to the country with the highest ratio of developed games to Github repositories, Austria is at first place with 25,33%. The only other country with a similarly high ratio is the Faroe Islands at exactly 25.00%. However, since only eight games were developed in the Faroe Islands, this number becomes less impressive. Denmark (22.12%), Slovenia (21.05%), El Salvador (20.83%) and Cambodia (20.00%) are the other countries that have a GitHub ratio of at least 20%. The global average ratio only sits at 11.40%

How is GitHub used during the Global Game Jame event? Using repositories for their Global Game Jam projects is not very common for jammers, but if they choose to use one then it is most likely on GitHub. The number of Global Game Jam projects that are hosted there also increases year over year. However, it is rare for jammers to actively commit to their repositories after a Global Game Jame event. Jammers that use Github seem to be the more involved with the Global Game Jam, having a higher degree and more attendances on averages than other jammers.

5.3. Summary

We analyzed data from the Global Game Jam events ranging from 2014 to 2018, which is getting more and more popular over the years. We first determined some basic information about the skills of the jammers like them having on average 3.35 skills, with the most popular ones being Programming, Game Design and Game Development. The rarest skills, on the other hand, are Hardware and Marketing. We also showed that most jammers only participated in one Global Game Jam event, the total average being 1,3764. Jammers that participate in more events typically also have

more skills and a higher degree. We found out that certain skills seem to lead jammers to collaborate with more people, meaning they could lead to a higher degree. The skills Audio and Music stand out here in particular. Jammers with these skills also participate in the development of more games per event than others. We also showed that the pure amount of skills a jammer possesses has nothing to do with their degree. For jammers that participate in multiple events we concluded based on the growth of their degree compared to their weighted degree that they tend to primarily work with new people at each event, the low amount of jammers that return to the event being a prime contributor to this fact.
Concerning technologies and platforms, we showed that jammers use a diverse array of tools. The variety has also steadily increased over the years. That said, the Unity engine and Windows platform are by far the most popular choices in nearly every country in the world. The only exceptions are countries with a relatively short Global Game Jam history.
We also took a look at the rare breed of the zero-degree jammers. While these particular jammers have on average more skills than the other participants, they especially stand out for using a very wide array of technologies and platforms. This may be their primary motivation for preferring to work on their own instead of teams. However, that lack of interaction seems also to be the reason that they are very unlikely to attend a Global Game Jam event again.
Finally, we took a look at how jammers utilized GitHub, the most popular repository provider among jammers. As our data shows, most jammers commit to the repositories of their projects only during the event. Only a very small subset of them still works on their project a year after the event. Jammers that use GitHub also sport a high degree, high average number of skills and a high average participation rate.

6. Example Applications of the Data

In this chapter we will give a short overview of the two applications that were made based on the collected data and the analysis.

6.1. Team Recommender

The Team Recommender is a simple web application that provides a means to arrange the jammers of a particular Jam Site roughly into equally skilled development teams for the Global Game Jam. The data for the Team Recommender is based on the results of the teams that still worked on their Global Game Jam project one year after the respective Global Game Jam ended. After entering the URL of the desired Jam Site, the application crawls the sites of all registered jammers and gathers their skill and if they previously attended a Global Game Jam. The framework the Team Recommender utilizes is Scrapy[1], which was also used to gather the general dataset in this work. Each skill and being a veteran jammer has a score assigned to it. These scores are based on the dataset. An overall score for each jammer is calculated utilizing these values. The formula for each skills score is:

$$Score_{\text{skill}} = \frac{\sum \frac{sm}{s}}{n} \tag{6.1}$$

where sm is the number of members in a team that possess the particular skill and s the total size of the team in question. n is the number of all motivated teams. It should, however, be noted that this formula is biased

[1] https://doc.scrapy.org

towards the skill sets of smaller teams. This results in the following point values seen in Table 6.1. As the average team size of the motivated teams

Skill	Score
2D Art	0.30516736
3D Art	0.18052105
Animation	0.15489993
Audio	0.14635956
Game Design	0.54496204
Game Development	0.53744824
Hardware	0.10751380
Marketing	0.05695307
Music	0.13853520
Programming	0.63946687
Project Management	0.26211180
Quality Assurance	0.15813492
Story and Narrative	0.24258109
Web Design	0.22974465
Writing	0.17548309
Veteran jammer	0.60346791

Table 6.1.: Score value of specific skill for Team Recommender

is 3.5580 (1.8997) with a median of three, the Team Recommender tries to build teams with three members. The resulting teams should each have roughly the same overall score. The approach here is similar to the Longest Processing Time algorithm. In detail, the algorithm starts with computing the optimal average team score of all jammers of a site. Next, it starts to create teams by grouping the jammer with the highest score with the jammer with the lowest score. Next comes the jammer with the second-highest score, who is matched with the jammer with the second-lowest score, and so on. After those steps, all teams have two members. The final team member is decided by iterating through all remaining jammers and picking the one that gets the team the closest to the previously computed optimal team average. If there are any jammers left after this process, they are assigned to the teams with the lowest scores.
One detail here is that in the case that the numbers of jammers is not

Figure 6.1.: Example output of the Team Recommender

divisible by three, the teams with four members will have a higher score than those with only three members. The screenshot in Figure 6.1 shows an example output based on the jam site of the TU Graz from 2018[2].

6.2. Visualization Dashboard

The Visualization Dashboard shows various charts that are based on some of the data from this work. The dashboard is implemented in Dash, a Phyton based framework developed by Plotly [3]. Plotly is based on Flask[4], Plotly.js[5] and React.js [6]. Flask is a WSGI web application framework. Plotly.js is used by the actual graphs and visualizations of the program and is an implementation of the d3 Javascript library[7]. React.js handles the interactive

[2]https://globalgamejam.org/2018/jam-sites/global-game-jam-graz
[3]https://dash.plot.ly
[4]https://palletsprojects.com/p/flask/
[5]https://plot.ly/javascript/
[6]https://reactjs.org/
[7]https://d3js.org/

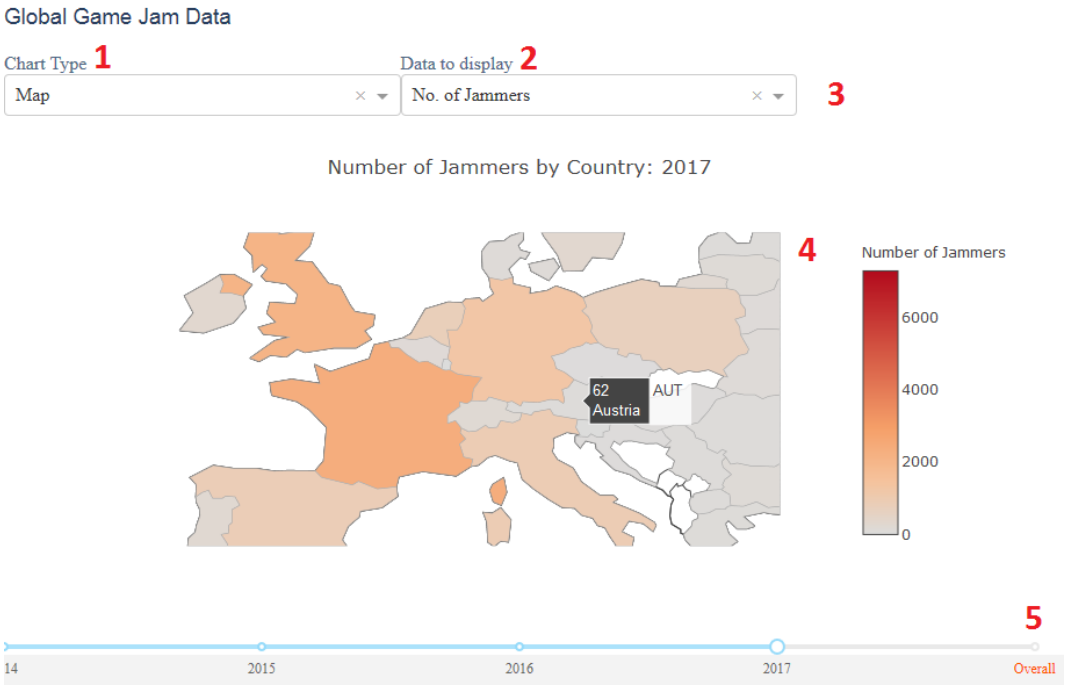

Figure 6.2.: Overview of the Visualization Dashboard

parts of the user interface. The data that the dashboard accesses for the visualizations are stored in the form of csv-files.

A short description of the elements shown in Figure 6.2

1. Chart style: Here the user can select which chart type to display. One can choose between line, bar, pie and map charts.
2. Data: The data to display can be chosen here. Only data that is appropriate for the given chart style will be shown.
3. Scale selector: For certain chart types, the user can select between a linear and logarithmic scale.
4. Chart: The main visualization. The user can hover over elements of the chart to display more details and also zoom in and out of the chart. A double-click will reset the view.
5. Detail selector: This will only be displayed on certain datasets where the user can select a specific country or year.

6.3. Summary

Based on the collected data and conducted analysis, we implemented two applications. The purpose of the first application, the Team Recommender is to simplify the team forming process at the start of a Global Game Jam event. It is based on the team composition of motivated Global Game Jam participants. For our purposes, that meant teams that still worked on their project on GitHub a year after the event. The usage of the program is simple: after entering the address of the desired jam site, the Team Recommender will scrape all registered members and their skills from the page. This information is used to create a list of teams consisting of three jammers each. The teams are supposed to be roughly equal in skill. The worth of each skill was determined by how common it is on average in the motivated teams. Having programming experience and participation in multiple events is considered especially valuable.
The second application is a simple visual dashboard, allowing users to view the collected data in the form of various chart types, like line graphs or map charts. The dashboard is implemented using the Dash Python framework, which includes Flask, Plotly.js, and React.js. The data for the dashboard is stored in csv-files.

7. Conclusion and Outlook

We will end the thesis here by discussing what further work can be done with our dataset and applications and what we learned during the process. Lastly, we will sum up the major points of this work.

7.1. Future Work

A major improvement that could be done in terms of data collection is to streamline the entire process. The data was gathered with several independent crawlers, that sometimes needed some data-reprocessing to receive the right input. Ideally, the whole process from crawling the data, updating the database and computing the necessary data for the next steps would be linked together, so that the whole process can be done once a year after a Global Game Jam event.
For this thesis, only a part of the collected data was used for the analysis, particularly concerning the GitHub data. For example, similar to Ortu et al. (2015), the GitHub commit messages could be analyzed to get a better understanding of the working climate and processes in the teams.
To improve the Team Recommender it would be helpful if jammers would at least state if their proficiency in any skill is at a beginner or already at a more advanced level. A real-world test of the application at an actual event would also be most welcome, including a post-event survey for the participants. The initial idea for the visualization dashboard was for it to be embedded into the Global Game Jam main site, to get any visitor of the page the opportunity to learn more about the event. We asked a person associated with the Global Game Jam event about this, but sadly never got a reply.

7.2. Lessons Learned

In regards to the crawling of the data, the main realization was that it is important to thoroughly analyze the origin websites as well as the data required. In the beginning, we naively crawled the games from the Global Game Jam Website as they were listed. However, the site presents the games in an arbitrary order, and even worse omits some of them. So our acquired game list always fell a few titles short of the supposed totals. This, in turn, led to much time spent on looking for an error in the crawler. Only after further analyzing the source code of the web page we discovered how the site is internally structured. This led to a more robust and complete crawling process. In regards to the analysis, being aware of the limitation of the dataset we had to work with was a key factor. For example, we wanted to look into the skillsets and connections of GitHub users in more detail. Sadly, with our data, we had no way to directly map jammers to GitHub users. To sidestep this issue, we instead took the whole team that worked on the game's repositories into consideration.

7.3. Summary

The development of any software project is a difficult task, and that is especially true for games. A good game does not only need technically versed programmers, but also skilled individuals from other fields, like writers and artists. But getting some first practical experience in the field can be tough. Taking part in a game jam event can be an opportunity to gain such experience. Through the various constraints and challenges that are inherent to a game jam event, the participants may get over their inhibitions and create new, innovative game ideas. All while acquiring new skills and making new friends along the way. In this thesis, we discussed Global Game Jam events to get a better understanding of their participants. We made use of data analysis and social network analysis to do so.
We first explained some basics about game development in general and what kind of skills are required to make a game. We then explained game jam events as an opportunity to get some first-hand experience in game

development. We listed popular game jam formats as well as the challenges and advantages they provide. We also discussed the importance of collaboration and version control in software engineering. We then showed how we acquired the relevant data from the Global Game Jam website. We collected data about the games, the responsible teams, and their skills. We used this data to construct a network as a foundation for the data analysis.
We used the degree of the jammers to explore how many connections they form during the event and if the skillset they posses affects that. We showed that on average a jammer has, regardless of skill set, a degree of 5.294. However, jammers that are skilled in Audio or Music tend to have a higher degree than the average. We have shown that this higher degree might stem from the fact that these jammers work on more games per event than other jammers. Jammers adept in Programming have the lowest average degree, while at the same time being the most common skill among jammers. Another factor that contributes to a higher degree is repeated participation in the event. Jammers that took part in the Global Game Jam event multiple times also posses more skills. Participating in a Global Game Jam event is a good opportunity to meet new people, as even veteran jammers tend to work with new team members. We argue that the low number returning jammers contributes to this.
Seeing as at the Global Game Jam event jammers are free to use any tools they like to develop their games, we also explored their utilized technologies and platforms. We showed that even though there are many different tools used, the majority sticks to Unity and MS Windows. Jammers that do not work in teams are particularly noteworthy for making use of some unusual tools. In regards to GitHub, we discussed how it is by far the most popular provider for project repositories, but at the same time those repositories are hardly used once the event is over.
Finally, we developed two small applications based on our data. The first is a dashboard to visualize the gathered data. It provides a multitude of datasets and chart types for interested users. The second application aims to simplify the team-building process at the start of the event. Based on the data of motivated teams we tried to build teams of three that are evenly skilled.

Appendix

Appendix A.

Tables

Country	Degree	Weighted	Jammers	Games	Repos	Sites
Afghanistan	5.8462	5.8462	13	2	0	1
Algeria	2.5094	2.8302	41	24	0	3
Angola	3.2222	3.2222	15	7	1	2
Argentina	8.3307	9.5285	1738	705	50	69
Armenia	4.4167	5.5278	61	18	1	2
Australia	6.9352	8.0288	1947	942	87	48
Austria	7.5654	7.9869	222	75	19	9
Bahrain	0.8571	0.8571	7	6	1	1
Belarus	5.2749	5.5945	228	85	7	4
Belgium	5.8623	6.4631	380	125	15	19
Bolivia	5.1525	5.9007	213	83	14	9
Brazil	8.1905	9.8872	5599	2098	175	240
Bulgaria	5.8044	6.6149	591	220	29	19

Table A.1.: Average degree, average weighted degree, number of jammers, number of games, number of repositories, and number of sites per country, A-B

Country	Degree	Weighted	Jammers	Games	Repos	Sites
Cambodia	3.4444	3.4444	18	5	1	1
Cameroon	2.4	2.4	10	3	0	1
Canada	5.6149	6.2432	4270	1699	272	122
Chile	6.775	7.3388	609	170	6	18
China	3.7985	4.0582	1429	647	29	29
Colombia	10.4894	11.7797	564	167	11	21
Costa Rica	4.8293	5.3854	144	63	4	8
Croatia	2.8039	2.902	47	18	2	2
Cuba	4.2165	5.3346	199	72	1	4
Czech Republic	4.5068	5.7466	93	56	8	6
Denmark	4.5796	4.9522	233	104	23	6
Dominican Rep.	3.1818	3.1818	22	7	1	1
Ecuador	5.5965	5.6754	94	27	3	5
Egypt	6.9725	7.8969	1559	524	16	8
El Salvador	2.8235	2.8235	16	6	0	1
Estonia	4.2941	4.4118	88	28	3	3
Ethiopia	12.0	12.0	13	1	0	1
Faroe Islands	5.75	5.75	37	8	2	2
Finland	6.3261	7.4295	2394	1041	167	99
France	6.5432	7.3369	4331	1446	207	123
Gabon	0.0	0.0	1	1	0	1
Georgia	4.0238	4.0238	39	13	0	2
Germany	6.5086	7.3888	2766	1035	201	112
Ghana	12.82	15.56	37	7	1	7
Greece	4.3503	5.3209	263	118	3	14
Guatemala	7.5333	8.5273	106	38	7	4
Guernsey	3.2273	4.25	29	21	1	4
Hong Kong	6.3359	7.4726	705	204	14	5
Hungary	5.0778	5.2778	67	28	4	5
Iceland	1.5714	1.5714	20	11	1	2
India	3.2279	3.4393	454	200	16	23
Indonesia	5.9879	6.5513	1190	416	19	50
Iran	4.8065	5.2375	272	96	1	32
Ireland	5.2548	5.9753	666	285	28	26
Israel	9.312	10.6832	918	404	53	25
Italy	9.2618	10.3945	2232	608	54	42

Table A.2.: Average degree, average weighted degree, number of jammers, number of games, number of repositories, and number of sites per country, C-I

Country	Degree	Weighted	Jammers	Games	Repos	Sites
Jamaica	4.1667	5.0	15	11	0	4
Japan	8.0742	8.3519	2055	532	38	98
Jordan	2.0472	2.3734	192	115	0	10
Kuwait	3.2063	3.381	38	29	0	3
Latvia	5.1509	5.9119	111	48	10	4
Lebanon	2.5469	2.6719	56	26	0	4
Lithuania	5.2422	6.1417	761	395	22	17
Luxembourg	4.8462	6.1462	97	43	0	4
Macao	3.3784	3.5946	32	12	0	2
Macedonia	6.4312	9.3039	259	99	11	8
Madagascar	4.25	4.25	8	2	0	1
Malaysia	6.7941	7.3462	309	118	8	9
Malta	7.382	9.4157	156	68	6	5
Mexico	8.9962	10.5819	2472	654	43	91
Moldova	2.1	2.1	29	12	1	1
Morocco	1.9091	1.9091	43	33	1	7
Nepal	3.5	5.55	31	13	1	3
Netherlands	7.5889	9.1936	2152	734	62	48
New Zealand	5.3648	6.2121	683	316	59	20
Nigeria	6.2121	6.8182	54	19	1	6
Norway	8.3361	9.3032	944	351	30	27
Pakistan	3.0294	3.1765	28	12	2	4
Palest. Territory	2.6714	2.6714	61	29	1	3
Panama	8.5893	9.9107	40	18	2	4
Paraguay	4.9794	5.433	76	26	4	3
Peru	7.7271	8.6667	340	111	13	11
Philippines	5.8003	7.0388	1310	490	20	23
Poland	5.2574	6.5736	1346	725	67	42
Portugal	4.8238	5.2554	611	231	16	26
Puerto Rico	3.25	3.25	68	25	1	3
Réunion	3.9583	5.2167	83	42	0	7
Romania	3.8115	4.2565	157	62	5	5
Russia	3.7277	4.2254	487	220	25	21

Table A.3.: Average degree, average weighted degree, number of jammers, number of games, number of repositories, and number of sites per country, J-R

Country	Degree	Weighted	Jammers	Games	Repos	Sites
Saudi Arabia	3.5447	3.5447	122	35	0	5
Serbia	4.5273	5.7109	168	90	10	8
Singapore	6.7411	7.8121	672	256	4	7
Slovakia	2.1538	2.6154	23	9	1	1
Slovenia	5.2308	5.6	46	19	4	4
South Africa	4.9661	5.375	241	149	7	16
South Korea	4.025	4.0938	399	170	8	28
Spain	6.5261	7.5669	2434	819	98	75
Sweden	5.1031	5.6984	871	349	57	43
Switzerland	6.1222	7.1318	426	170	25	16
Taiwan	6.91	7.309	597	159	13	20
Thailand	5.445	6.3786	556	185	16	7
Togo	0.0	0.0	1	1	0	1
Trinidad & Tob.	0.0	0.0	1	1	0	1
Tunisia	3.2654	3.5329	412	185	1	15
Turkey	6.3347	7.2848	1288	606	48	40
Uganda	0.0	0.0	3	3	0	1
Ukraine	4.1188	4.5022	377	136	20	7
United Arab Em.	4.9884	5.2907	64	27	1	7
United Kingdom	5.5443	6.356	5380	2157	252	201
United States	7.2992	8.0872	17551	6834	1066	698
Uruguay	11.8784	12.9082	248	66	6	11
Venezuela	8.3766	10.1563	380	147	25	13
Vietnam	2.3298	2.3298	87	49	2	4
Zambia	3.566	3.6604	37	22	2	5

Table A.4.: Average degree, average weighted degree, number of jammers, number of games, number of repositories, and number of sites per country, S-Z

Bibliography

Asuncion, H., Socha, D., Sung, K., Berfield, S., & Gregory, W. (2011). Serious game development as an iterative user-centered agile software project. In *Proceedings of the 1st international workshop on games and software engineering* (pp. 44–47). ACM.

Bay, J. W. (2018). What does a video game writer do, and how can i start in that career? Retrieved August 23, 2019, from https://www.gameindustrycareerguide.com/what-does-a-video-game-writer-do/

Bethke, E. (2003). *Game development and production*. Wordware Publishing, Inc.

Beynon-Davies, P., Carne, C., Mackay, H., & Tudhope, D. (1999). Rapid application development (rad): an empirical review. *European Journal of Information Systems*, *8*(3), 211–223.

Borg, M., Garousi, V., Mahmoud, A., Olsson, T., & Stalberg, O. (2019). Video game development in a rush: a survey of the global game jam participants. *IEEE Transactions on Games*.

Borgatti, S. P., Mehra, A., Brass, D. J., & Labianca, G. (2009). Network analysis in the social sciences. *science*, *323*(5916), 892–895.

Cohn, M. (2008). Agile and scrum for video game development. Retrieved August 23, 2019, from https://www.mountaingoatsoftware.com/presentations/agile-and-scrum-for-video-game-development

Cook, M., Smith, G., Thompson, T., Togelius, J., & Zook, A. (2015). Hackademics: a case for game jams at academic conferences. In *Proceedings of the 10th international conference on the foundations of digital games, pacific groove, ca, usa*.

De Alwis, B. & Sillito, J. (2009). Why are software projects moving from centralized to decentralized version control systems? In *2009 icse workshop on cooperative and human aspects on software engineering* (pp. 36–39). IEEE.

Deen, M., Cercos, R., Chatman, A., Naseem, A., Bernhaupt, R., Fowler, A., ... Mueller, F. (2014). Game jam:[4 research]. In *Chi'14 extended abstracts on human factors in computing systems* (pp. 25–28). ACM.

Ferrara, F. (2018). Agile game development process for indies. Retrieved September 22, 2019, from https://www.gamasutra.com/blogs/FabioFerrara/20181218/333112/Agile_Game_Development_process_for_Indies.php

Fowler, A., Khosmood, F., & Arya, A. (2013). The evolution and significance of the global game jam. In *Proc. of the foundations of digital games conference* (Vol. 2013).

Fowler, A., Khosmood, F., Arya, A., & Lai, G. (2013). The global game jam for teaching and learning. In *Proccedings of the 4th annual conference on computing and information technology research and education new zealand* (pp. 28–34).

Fowler, A., Lai, G., Studios, K., & Khosmood, F. (2015). Trends in organizing philosophies of game jams and game hackathons. In *Gj workshop. fdg2015*.

Ghahrai, A. (2018). Rapid application development (rad). Retrieved July 10, 2019, from https://www.testingexcellence.com/rapid-application-development-rad/

Global Game Jam. (2014). Globalgame jam theme 2014. Retrieved May 3, 2019, from https://globalgamejam.org/news/ggj14-theme

Goddard, W., Byrne, R., & Mueller, F. (2014). Playful game jams: guidelines for designed outcomes. In *Proceedings of the 2014 conference on interactive entertainment* (pp. 1–10). ACM.

Hanneman, R. A. & Riddle, M. (2005). Introduction to social network methods. http://faculty.ucr.edu/~hanneman/nettext/index.html. University of California Riverside.

Hecker, C., Ulrich, T., & Hall, J. (2004). Indie game jam. Retrieved July 1, 2019, from http://www.indiegamejam.com/

Hirst, T. (2008). The process of game creation & the game design document. Retrieved August 13, 2019, from https://digitalworlds.wordpress.com/2008/04/10/the-process-of-game-creation-the-game-design-document/

Jarczyk, O., Gruszka, B., Jaroszewicz, S., Bukowski, L., & Wierzbicki, A. (2014). Github projects. quality analysis of open-source software. In *International conference on social informatics* (pp. 80–94). Springer.

Juego Studios. (2017). What is the game development life cycle? Retrieved August 13, 2019, from https://www.gamasutra.com/blogs/JuegoStudios/20170413/296019/What_Is_the_Game_Development_Life_Cycle.php

Kanode, C. M. & Haddad, H. M. (2009). Software engineering challenges in game development. In *2009 sixth international conference on information technology: new generations* (pp. 260–265). IEEE.

Kultima, A. (2015). Defining game jam. In *Fdg*.

Laniado, D., Tasso, R., Volkovich, Y., & Kaltenbrunner, A. (2011). When the wikipedians talk: network and tree structure of wikipedia discussion pages. In *Fifth international aaai conference on weblogs and social media*.

Law, A. & Charron, R. (2005). Effects of agile practices on social factors. In *Acm sigsoft software engineering notes* (Vol. 30, 4, pp. 1–5). ACM.

Law, B. & McDonald, B. (2015). Game jams: how can they influence software development curricula. In *Workshop on game jams, hackathons and game creation events, pacific grove, ca, usa*.

Lightsey, B. (2001). *Systems engineering fundamentals*. DEFENSE ACQUISITION UNIV FT BELVOIR VA.

Lima, A., Rossi, L., & Musolesi, M. (2014). Coding together at scale: github as a collaborative social network. In *Eighth international aaai conference on weblogs and social media*.

Ludum Dare. (2017). Ludum dare. Retrieved June 25, 2019, from https://ldjam.com

Marcelo Madril, A. (2016). Lead programmer: less programming, more humaning. Retrieved September 22, 2019, from https://www.gamasutra.com/blogs/ArielMarceloMadril/20160114/263585/Lead_Programmer_less_programming_more_humaning.php

Martin, J. (1991). *Rapid application development*. Macmillan Publishing Co., Inc.

Meriläinen, M. & Aurava, R. (2018). Internal barriers to entry for first-time participants in the global game jam. In *12th european conference on games based learning* (pp. 414–421).

Mullich, D. (2015). Roles on a game development team. Retrieved August 23, 2019, from https://davidmullich.com/2015/02/02/roles-on-a-game-development-team/

Musil, J., Schweda, A., Winkler, D., & Biffl, S. (2010). Synthesized essence: what game jams teach about prototyping of new software products.

In *Proceedings of the 32nd acm/ieee international conference on software engineering-volume 2* (pp. 183–186). ACM.

Nordic Game Jam. (2019). Nordic game jam. Retrieved June 30, 2019, from https://www.nordicgamejam.com/

Ortu, M., Destefanis, G., Adams, B., Murgia, A., Marchesi, M., & Tonelli, R. (2015). The jira repository dataset: understanding social aspects of software development. In *Proceedings of the 11th international conference on predictive models and data analytics in software engineering* (p. 1). ACM.

Otte, E. & Rousseau, R. (2002). Social network analysis: a powerful strategy, also for the information sciences. *Journal of information Science*, *28*(6), 441–453.

Oxagile. (2014). Waterfall software development model. Retrieved June 15, 2019, from https://www.oxagile.com/article/the-waterfall-model

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The pagerank citation ranking: bringing order to the web.

Peterson, K. (2013). The github open source development process. *url: http://kevinp. me/github-process-research/github-processresearch. pdf (visited on 05/11/2017)*.

Petrillo, F., Pimenta, M., Trindade, F., & Dietrich, C. (2009). What went wrong? a survey of problems in game development. *Computers in Entertainment (CIE)*, *7*(1), 13.

Pirker, J., Economou, D., & Gütl, C. (2016). Interdisciplinary and international game projects for creative learning. In *Proceedings of the 2016 acm conference on innovation and technology in computer science education* (pp. 29–34). ACM.

Pirker, J., Khosmood, F., & Gütl, C. (2017). Social network analysis of the global game jam network. In *Proceedings of the second international conference on game jams, hackathons, and game creation events* (pp. 10–14). ACM.

Pirker, J., Lesjak, I., Punz, A., & Drachen, A. (2018). Social aspects of the game development process in the global gam jam. In *Proceedings of the international conference on game jams, hackathons, and game creation events* (pp. 9–16). ACM.

Pirker, J. & Voll, K. (2015). Group forming processes-experiences and best practice from different game jams. In *Workshop proceedings of the 10th international conference on the foundations of digital games (pacific grove, california, asilomar conference grounds)*.

Prell, C. (2012). *Social network analysis: history, theory and methodology*. Sage.

Reng, L., Schoenau-Fog, H., & Kofoed, L. B. (2013). The motivational power of game communities-engaged through game jamming. In *Proceedings of the 8th international conference on the foundations of digital games* (pp. 14–17).

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on software engineering* (pp. 328–338). IEEE Computer Society Press.

Sanneblad, J. & Holmquist, L. E. (2003). Prototyping mobile game applications. In *Entertainment computing* (pp. 117–124). Springer.

Schwaber, K. & Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, *21*, 19.

Shyleno, P. (2019). Understanding the roles of game dev professionals. Retrieved August 23, 2019, from https://www.gamasutra.com/blogs/PavelShylenok/20190115/334322/Understanding_the_Roles_of_Game_Dev_Professionals.php

Slavic Game Jam. (2018). Slavic game jam. Retrieved July 1, 2019, from https://slavicgamejam.org/

Takeuchi, H. & Nonaka, I. (1986). The new new product development game. *Harvard business review*, *64*(1), 137–146.

Tyler, J. R., Wilkinson, D. M., & Huberman, B. A. (2005). E-mail as spectroscopy: automated discovery of community structure within organizations. *The Information Society*, *21*(2), 143–153.

van Kelle, E., Visser, J., Plaat, A., & van der Wijst, P. (2015). An empirical study into social success factors for agile software development. In *2015 ieee/acm 8th international workshop on cooperative and human aspects of software engineering* (pp. 77–80). IEEE.

Warton, S. (2015). 48hr game making competition. Retrieved August 5, 2019, from http://48hrgamecomp.com/about/

Wasserman, S., Faust, K. et al. (1994). *Social network analysis: methods and applications*. Cambridge university press.

Watson, A. (2018). Film industry - statistics & facts. Retrieved August 13, 2019, from https://www.statista.com/topics/964/film/

Wearn, N. & McDonald, B. (2016). Ethos of location and its implication to the motivators of global games jam participants. In *Proceedings of the international conference on game jams, hackathons, and game creation events* (pp. 58–61). ACM.

Wijman, T. (2019). The global games market will generate $152.1 billion in 2019 as the u.s. overtakes china as the biggest market. Retrieved August 13, 2019, from https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/

Zook, A. & Riedl, M. O. (2013). Game conceptualization and development processes in the global game jam. In *Workshop proceedings of the 8th international conference on the foundations of digital games* (pp. 1–5).