How can game developers leverage data from online distribution platforms? A case study of the Steam platform

by

Dayi Lin

A thesis submitted to the School of Computing

in conformity with the requirements for the

Degree of Doctor of Philosophy

Queen's University

Kingston, Ontario, Canada

January 2019

Abstract

DEVELOPING a successful game is challenging. Prior work shows that gamers are extremely difficult to satisfy, making the quality of games an important issue. Prior work has yielded important results from mining data that is available on the online distribution platforms for software applications, helping practitioners save valuable resources, and improving the user-perceived quality of software that is distributed through these platforms. However, much of the work on mining online distribution platforms focuses on mining mobile app stores (e.g., Google Play Store, Apple App Store). Video game development differs from the development of other types of software. Hence, knowledge derived from mining mobile app stores may not be directly applicable to game development.

In this Ph.D. thesis, we focused on mining online distribution platforms for games. In particular, we mined data from the Steam platform, the largest digital distribution platforms for PC gaming, with over 23,000 games available and over 184 million active users. More specifically, we analyzed the following four aspects of online distribution for games: urgent updates; the early access model (which enables game developers to sell unfinished versions of their games); user reviews; and user-recorded gameplay videos on the Steam platform. We observed that the choice of update strategy is associated with the proportion of urgent updates that developers have to release. Early access game developers can use the early access model as a method for eliciting early feedback and more positive reviews to attract additional new players. In addition, although negative reviews contain more valuable information for developers, the portion of useful information in positive reviews should not be ignored by developers and researchers. Finally, we proposed an approach for determining the likelihood that a gameplay video demonstrates a game bug, with both a mean average precision at 10 and a mean average precision at 100 of 0.91. Our approach can help game developers leverage readily available gameplay videos to automatically collect otherwise hard-to-gather bug reports for games. The results of our empirical studies highlight the value of mining online distribution platforms for games in offering practical suggestions for game developers.

Acknowledgments

I am extremely grateful for the supervision I received from Prof. Ahmed E. Hassan. Without his continuous guidance and support, this thesis would not be possible.

I would like to thank Prof. Cor-Paul Bezemer for his patient help and guidance throughout my graduate study. I would also like to thank my co-authors Prof. Ying Zou and Prof. Chakkrit Tantithamthavorn for their contributions and feedback in the studies. In addition, I appreciate Prof. Nicholas Graham, Prof. Hossam Hassanein, Prof. Mohammad Zulkernine, Prof. Kai Salomaa, Prof. Robin Dawes, Prof. Yuan Tian, Prof. Ali Etemad, and Prof. Ali Ouni for the advice that they gave me.

I thank Prof. Jennifer R. Whitson from the University of Waterloo and Mr. Patrick Walker from EEDAR for the inspiring discussions about the gaming industry, and Sergey Galyonkin, the owner of Steam Spy, who generously gave us access to all the historical data of Steam as collected by Steam Spy for my research. I am also grateful for the attention and feedback that I received from the gaming community. In particular, I appreciate Lars Doucet for introducing me to the indie game developer community, and providing me with their data.

My appreciation extends to my fellow labmates and alumni, visiting researchers, and friends from the Software Analysis and Intelligence Lab (SAIL), including Prof. Tse-Hsun (Peter) Chen, Prof. Weiyi Shang, Prof. Yasutaka Kamei, Prof. Daniel Alencar da Costa, Dr. Gustavo Ansaldi Oliva, Dr. Mark D. Syer, Dr. Mohamed Sami Rakha, Dr. Shaowei Wang, Daniel Lee, Gopi Krishnan Rajbahadur, Heng Li, Jiayuan Zhou, Safwat Hassan. I also thank my friends Chenyang Song, Chunyang Zhu, David Cox, Hao Li, Huixiang Huang, Michal Pasternak, Wenbo Liu, Xiaohe Zhang, for all the memories we shared in the journey.

I would also like to thank Prof. Xiaoge Li from Xi'an University of Posts and Telecommunication, who inspired me to pursue graduate studies.

Special thanks to my father and mother, Huijian Lin and Wenzhi Chen, who have been offering me their unconditional love throughout my entire life. I dedicate this thesis to my father and mother.

Co-authorship

In all chapters and related publications of the thesis, my contributions are: drafting the initial research idea; researching background knowledge and related work; collecting data; conducting experiments and data analysis; and writing and polishing drafts.

Earlier versions of the work in the thesis were published as listed below:

1. Studying the urgent updates of popular games on the Steam platform.
   Dayi Lin, Cor-Paul Bezemer and Ahmed E. Hassan. Empirical Software Engineering Journal (EMSE), 2017, 22(4), pp.2095-2126.

2. An empirical study of early access games on the Steam platform.
   Dayi Lin, Cor-Paul Bezemer and Ahmed E. Hassan. Empirical Software Engineering Journal (EMSE), 2018, 23(2), pp.771-799.

3. An empirical study of game reviews on the Steam platform.

   Dayi Lin, Cor-Paul Bezemer, Ying Zou and Ahmed E. Hassan. Empirical Software Engineering Journal (EMSE), 2018, In Press.

Statement of Originality

I, Dayi Lin, hereby declare that I am the sole author of this thesis. All ideas and inventions attributed to others have been properly referenced. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

THE steadily increasing popularity of computer games has led to the rise of a multi-billion dollar industry. With the revenue of the gaming industry reaching $91 billion in 2016 (SuperData, 2016), PC gaming is expected to grow at a rate of 6.3% annually through 2020 (Takahashi, 2016). The scale of this industry is demonstrated by the number of players which ranges to almost 900,000 players per day for popular games such as the *Dota 2* game (Gray, 2016).

The wide-spread availability of increasingly fast Internet connections has opened up a range of new opportunities for game developers, such as subscription-based gaming and a shifting of distribution strategies from offline physical distribution (e.g., through brick-and-mortar stores) to online digital distribution (e.g., through the Xbox Game Store (Microsoft, 2015) or Steam (Valve, 2016b)). Game purchases on online distribution platforms reached a revenue of $61 billion in 2015 (SuperData, 2018). Such online distribution platforms offer Digital Rights Management (DRM) service for developers, and serve as a storefront that allows users to purchase, download, and update their games. In addition, many platforms allow users to interact with developers and other users through features such as user reviews. The information contained within these distribution platforms is valuable for many technical, customer, and business aspects of games (Harman et al., 2012).

Prior work has yielded important results from mining data that is available on the online distribution platforms for software applications, helping practitioners save valuable resources, and improve the user-perceived quality of software that is distributed through these platforms (Vasa et al., 2012; Hoon et al., 2012; Pagano and Maalej, 2013). However, much of the prior work focuses on mining mobile app stores (e.g., Google Play Store, Apple App Store). Video game development differs from the development of other types of software (Pascarella et al., 2018a). Hence, knowledge derived from mining mobile app stores may not be directly applicable to game development.

1.1 Thesis Statement

In this Ph.D. thesis, we focused on mining online distribution platforms for games. In particular, we mined data from the Steam platform. The reason for focusing on the Steam platform is that Steam is considered as the largest digital distribution platforms for PC gaming, with over 23,000 games available and over 184 million active users (Galyonkin, 2018). More specifically, we analyzed the following four aspects of online distribution for games: urgent updates; the early access model[1]; user reviews; and user-recorded gameplay videos. We believe that the data on the Steam platform contains valuable information that can help developers produce games with better user-perceived quality. Through the mining of the data on the Steam platform we can provide practical suggestions for game developers. Therefore, we propose the following thesis statement:

Thesis Statement: Online distribution platforms (e.g., the Steam platform) contain valuable information that can yield important findings of game development, which can in turn help provide practical suggestions for game developers.

1.2 Thesis Overview

In this Ph.D. thesis, we first describe the background of the Steam platform (Chapter 2). Then, we present the state-of-the-art literature relevant to mining online distribution platforms for games (Chapter 3). We then detail each one of the four studied aspects of the online distribution for games: 1) urgent updates; 2) the early access model; 3) user reviews; and 4) user-recorded gameplay videos on the Steam platform. Lastly, we conclude the thesis (Chapter 8). Below, we briefly summarize the four chapters of our study on the four aspects of the online distribution for games.

[1]Early access is a model that allows customers to purchase the public beta version of a game while its developers continue working on the game.

1.2.1 Studying urgent updates of popular games on the Steam platform (Chapter 4)

Online distribution eases the continuous and rapid updating of released games. Urgent updates are updates that are deemed critical enough to not be left unreleased until an upcoming regular-cycle update. As urgent updates are usually released in a state of emergency, urgent updates cause unnecessary stress on developers. The stress of these so-called "fire-fighting conditions" can not only lead to inefficient problem solving, but also introduce changes that can easily create new problems (Bohn, 2000). We studied urgent updates of games on the Steam platform to understand the causes behind urgent updates. We observed that feature malfunctions, crashing games and visual bugs are the most commonly given reasons for releasing urgent updates. In addition, We observed that games that release frequently also release a higher proportion of 0-day updates than games that use a traditional build-up candidate update strategy.

1.2.2 Studying early access games on the Steam platform (Chapter 5)

Early access is a model that allows customers to purchase the public beta version of a game while developers continue working on the game. The early access model made a name for itself through several successful games, such as the *DayZ* game (EuroGamer, 2014). However, the benefits of the early access model have been questioned as well. For instance, the *Spacebase DF-9* game abandoned the early access stage unexpectedly as the raised funds during that early access stage were insufficient to continue its development, leading the game to receive 77% negative reviews (Valve, 2018). In order to get a better understanding of the impact and limitations of the early access model, we analyzed the early access games on the Steam platform to provide suggestions for developers to make optimal use of this novel release strategy. We observed that the percentage of players that review a game during its early access stage is lower than the percentage of players that review a game after leaving the early access stage. However, the average rating of the reviews is much higher during the early access stage. While the early access model is not a fix for low-quality games, the early access model appears to be a valuable tool for developers that want to improve their games by interacting with their players.

1.2.3 Studying game reviews on the Steam platform (Chapter 6)

Many online game distribution platforms allow users to post reviews of a game. These game reviews provide a rich data source that can be leveraged to better understand user-reported issues. Prior work on mobile app reviews has shown the value of studying reviews (Vasa et al., 2012; Hoon et al., 2012; Pagano and Maalej, 2013). However, the majority of recent research on the quality of games focuses on quality issues from the perspective of *developers* (Washburn Jr et al., 2016; Lewis et al., 2010), while few studies are related to the particular issues that *users* face when playing games. We studied the reviews on the Steam platform, to help game developers better understand how to leverage user reviews for improving the user-perceived quality of their games. We observed that players complain more about game design than bugs in their reviews. In addition, game reviews are different from mobile app reviews along several aspects. We suggested that information that can be extracted from positive reviews should not be ignored by future studies.

1.2.4 Studying gameplay videos on the Steam platform (Chapter 7)

A common practice to ensure the user-perceived quality of games is to allow players to submit bug reports to developers. Many studios encourage players to submit relevant screenshots or videos along with their bug report (Sixen, 2010; Afic, 2015; Habakuk, 2017), especially for graphic bugs ("should probably always have a screenshot" (Sixen, 2010)) and general functionality bugs (which "may only be visible in movies" (Sixen, 2010)). However, a prior study (Zimmermann et al., 2010) pointed out that it is difficult for bug reporters to collect supplemental materials such as videos. On the other hand, gameplay videos around game bugs have become popular on the Internet (Petrova and Gross, 2017), opening a new opportunity for developers to collect intuitive information of bugs. We analyzed the gameplay videos on the Steam platform to help developers leverage the videos and save valuable quality assurance resources. We observed that the naïve approach of identifying bug videos (such as keyword searching) is inefficient and imprecise. We then proposed an approach that uses the metadata of gameplay videos to train a random forest classifier, to determine the likelihood that a gameplay video showcases a game bug. Overall, the approach achieves a precision that is 43% higher than the precision of the naïve keyword searching approach. In addition, our approach achieves both a mean average precision at 10 and a mean average precision at 100 of 0.91, and shows a good generalizability.

1.3 Thesis Contribution

In this thesis, we studied four aspects of the online distribution for games through mining the data from a large online distribution platform for games, the Steam platform.

The results of our empirical studies highlight the value of mining online distribution platforms for games in offering practical suggestions for game developers. In particular, the thesis contributions are as follows:

1. We show that the choice of update strategy is associated with the proportion of urgent updates that game developers have to release.
2. We demonstrate the value of the early access model as a method for eliciting early feedback and more positive reviews to attract additional new players.
3. We show that although negative reviews contain more valuable information for developers, the portion of useful information in positive reviews should not be ignored by developers and researchers.
4. We propose an approach for determining the likelihood that a gameplay video showcases game bugs, with both a mean average precision at 10 and a mean average precision at 100 of 0.91. Our approach can help game developers leverage readily available gameplay videos to automatically collect otherwise hard-to-gather bug reports for games.

CHAPTER 2

Background on the Steam Platform

In this chapter, we provide a brief background of the Steam platform.

Steam is a digital game distribution platform, developed by Valve Corporation. Steam is considered as one of the largest digital distribution platforms for PC gaming, with over 23,000 games available and over 184 million active users (Galyonkin, 2018). Table 2.1 shows a comparison between the number of games on Steam and on several other PC gaming distribution platforms. Steam offers digital rights management (DRM), multiplayer gaming, and social networking services, through two major components of the Steam platform: the Steam Store (Valve, 2016b), and the Steam Community (Valve, 2018).

Users can purchase games from the Steam Store. Games purchased through the Steam Store, along with games purchased from third-party vendors and activated

Table 2.1: Comparison between the number of games on Steam and on other PC gaming distribution platforms (as of Dec 19, 2017)

Platform	**Number of PC Games**
Steam	18,711
Green Man Gaming[1]	5,978
GamersGate[2]	5,921
Good Old Games (GOG)[3]	2,232
Direct2Drive[4]	1,552
GameStop[5]	1,103
Origin[6]	318

[1] `https://www.greenmangaming.com`
[2] `https://www.gamersgate.com/`
[3] `https://www.gog.com/`
[4] `https://www.direct2drive.com/`
[5] `http://www.gamestop.com/`
[6] `https://www.origin.com/`

through the Steam platform, are playable for a user after logging in on Steam using the Steam client. The Steam client will verify ownership of the game and automatically install any available updates. It is mandatory to install the latest update of a game in order to play a game through Steam. As a result, players are always using the same version of a game, even if the last update was a buggy update. There is no option for undoing or skipping an update in Steam games.

In addition, users can enjoy social network-like features such as friends lists and chat functionality through the Steam Community. The Steam Community publishes statistics for games and players. Game developers and journalists can publish news updates for games on so-called channels. Table 2.2 lists all available channels with a brief description of the content of each channel. Various third-party dashboards, such as SteamSpy (Galyonkin, 2018), collect a plethora of aggregated information from the Steam Community about Steam games.

Table 2.2: All available Steam channels

Channel	**Contents**	**Used in our study**
Announcements	General updates including promotions	
Client Updates	Steam Client updates	
Eurogamer	Reviews of games	
Kotaku	Reviews of games	
Left 4 Dead Official Blog	Updates for the *Left 4 Dead* game	
PC Gamer	Reviews of games	
Portal 2 Official Blog	Updates for the *Portal 2* game	
Press Releases	Press releases for Valve games	
Product Releases	New game releases	✓
Product Updates	Game updates	✓
Rock, Paper, Shotgun	Reviews of games	
Shacknews	Reviews of games	
Steam Blog	General updates including promotions	
Steam Community Announcements	Updates for games and promotions	✓
TF2 Official Blog	Updates for the *Team Fortress 2* game	

In general, developers post announcements about game updates to one or more channels, e.g., to the *Product Update* channel. However, while installing the latest game update on Steam is mandatory for users, developers do not necessarily need to announce all updates that they make. Instead, they may choose to silently update a game. Nevertheless, developers do often post news updates about their games to keep users informed about the latest news about their games.

The Steam Community also permits users to post reviews of games once they played them. Different from other popular application distribution platforms which use a 5-star rating system for reviews, players are asked to provide their overall feeling about the game: “Recommended” (i.e., a positive review), or “Not Recommended” (i.e., a negative review). The number of playing hours of the reviewed game, the number of played games, and the number of previously posted reviews by the reviewer at this moment are shown alongside the review. The positive review rate ($\frac{\#\ of\ recommended\ reviews}{\#\ of\ all\ reviews}$) is displayed on the Steam Store page of the game, to advise potential customers. A user can only provide one review of a game, across all versions of the game. The user is allowed to update the review at a later time.

In addition to the review mechanism, the Steam Community provides a discussion forum for each game in which players and developers can communicate. The forum of a game can have a variety of subforums that are created by developers. By default, a forum contains two subforums that are created by Steam, which are *General Discussions* and *Trading*. The *Trading* subforum is specifically for players to trade in-game properties, such as rare weapons, while *General Discussions* normally contains threads regarding bug reports, suggestions, questions, etc.

In order to publish a game in the Steam Store, developers need to undergo a tax and identity verification process and pay a product submission fee of $100 for each of their games. In addition, the game must go through review periods where Steam personnel play each game to check that it is configured correctly, matches the description that is provided on the store page, and does not contain malicious content (Alden, 2017). The strict process of publishing a game on the Steam Platform ensures the quality of the games that are available on the Steam Store.

Mobile app stores, such as the Apple App store and the Google Play store, have similar review processes. However, compared to the Apple App Store, which requires an annual developer membership fee of $99 (Inc., 2017), or Google Play which has a one-time membership fee of $25 (Google, 2017), Steam requires a submission fee for each product submission.

CHAPTER 3

Literature Survey

IN this chapter, we first explain our literature selection process, then we discuss the related work along different categories.

3.1 Literature Selection

A large number of prior studies have focused on video games (e.g., serious games, which are designed for a primary purpose other than pure entertainment, such as education). However, our literature survey focuses on the studies that aim to understand and support online distribution and software engineering practices of games. Hence, our literature survey starts from papers that are published on major software engineering, as well as game journals and conferences. Table 3.1 lists venues from which we

Table 3.1: Names of conferences and journals as starting venues of the literature survey

Venue Type	Venue Name	Abbreviation
Journal	IEEE Transactions on Software Engineering	TSE
Journal	ACM Transactions on Software Engineering and Methodology	TOSEM
Journal	Empirical Software Engineering	EMSE
Journal	Automated Software Engineering	ASE
Journal	Journals of Systems and Software	JSS
Journal	Entertainment Computing	EC
Journal	Computers in Entertainment	CIE
Journal	Computer Supported Cooperative Work	CSCW
Conference	European Software Engineering Conference / ACM SIGSOFT Symposium on the Foundations of Software Engineering	ESEC/FSE
Conference	International Conference on Software Engineering	ICSE
Conference	International Conference on Automated Software Engineering	ASE
Conference	International Conference on Software Maintenance and Evolution	ICSME
Conference	International Conference on Software Analysis, Evolution, and Reengineering	SANER
Conference	International Conference on Mining Software Repositories	MSR
Conference	Foundation of Digital Games	FDG
Conference	International Conference on Entertainment Computing	ICEC
Conference	International Workshop on Games and Software Engineering	GAS
Conference	Conference on Computer Supported Cooperative Work & Social Computing	CSCW
Conference	Conference on Human Factors in Computing Systems	CHI
Conference	Annual Symposium on Computer-Human Interaction in Play	CHI PLAY

started our literature survey. We focus our survey on papers that were published in the last 10 years (i.e., from 2009 to 2018). We first read the title and abstract of each paper in the venues to include papers relevant to the categories of *mining online distribution platforms* and *software engineering and games*. Then we check the citations of each selected paper, to include relevant papers that were not published in the starting venues. We detail each category of papers below.

3.2 Mining Online Distribution Platforms

The area of mining online distribution platforms has attracted a large amount of attention from the Software Engineering community. Most of the work in this area focuses on mobile app stores.

3.2.1 Mining Mobile App Stores

Harman et al. (2012) introduced app store mining by analyzing paid apps in the Blackberry app store, and showed that app store mining can be valuable to app developers. Ruiz et al. (2012) conducted a large-scale case study on Android Market and provided an initial insight into the development practices of mobile apps. Martin et al. (2017) surveyed the field of app store analysis for software engineering. They observed an increasing scale of studied app samples and a diverse set of techniques and applications in app store analysis, highlighting the health and future potential of the field.

Mining mobile app reviews. Most of the studies on mobile app store analysis focus on mobile app reviews. Vasa et al. (2012) and Hoon et al. (2012) analyzed user reviews of mobile apps and observed that when users give a negative review to an app, the length of the feedback is greater. Pagano and Maalej (2013) observed that the quality and constructiveness of mobile app reviews vary widely, from helpful advice and innovative ideas to insulting and offensive comments. Palomba et al. (2015, 2018) surveyed 73 developers and observed that over 75% of them take reviews into account often, and that addressing user reviews can increase the overall user rating of an app.

Many studies focus on automatically extracting useful information from mobile app reviews. Fu et al. (2013) proposed a system called WisCom to analyze mobile app reviews at the market, app, and review level. Chen et al. (2014) presented “AR-Miner”, a framework to extract and prioritize informative reviews for developers. Palomba et al. (2017) introduced an approach to not only extract feedback from reviews, but also link such feedback back to source code components. Iacob and Harrison (2013) proposed MARA, a prototype for the automated retrieval of mobile app feature requests from online reviews. Di Sorbo et al. (2016) introduced an approach to summarize recommended software changes from user reviews. Villarroel et al. (2016), Scalabrino et al. (2017), and Ciurumelea et al. (2017) developed techniques to automatically cluster and prioritize reviews for better release planning. Gao et al. (2018) proposed a framework to identify emerging app issues. Wei et al. (2017) leverages user reviews to prioritize the warnings from Lint, the static analyzer.

Genc-Nayebi and Abran (2017) systematically reviewed literature about opinion mining from mobile app store user reviews. Martens and Johann (2017) performed an exploratory study of the emotional sentiment of seven million reviews from the Apple App Store. Vu et al. (2015a,b) proposed a keyword-based approach that automatically extracts keywords and ranks their associations with negative reviews. Vu et al. (2016) later proposed another phrase-based approach to monitor the outbreaks of negative sentiments over time. Khalid et al. (2014) observed that some devices tend to generate worse ratings than others, and suggested developers to focus their test resources on the devices with the most negative reviews. Noei et al. (2017) studied the relation of both device attributes and app attributes with the user-perceived quality of Android apps, and observed that the device attributes (such as the CPU) also have a strong association with the user ratings.

Several studies focus on the categorization of mobile app reviews. Khalid et al. (2015) manually identified 12 types of complaints that users raise in their reviews.

McIlroy et al. (2016b) studied the multi-labelled nature of reviews from 20 mobile apps, and proposed an automatic multi-labelling approach for mobile app reviews. Gu and Kim (2015) presented a framework to classify reviews into five categories and used a pattern-based parser to extract software aspects. Panichella et al. (2015) presented a taxonomy to classify app reviews as well as an approach to automatically classify app reviews to the proposed categories. Guzman et al. (2015) evaluated the performance of individual machine learning algorithms and their ensembles for app review classification, and concluded that the ensembles outperform the individual classifiers.

Several studies focus on the ratings that are posted with user reviews. Ruiz et al. (2016) examined more than 10,000 unique mobile apps in the Google Play store and stated that due to the evolving nature of mobile apps, the current displayed score generated from reviews is not dynamic enough to show the changing user satisfaction level. Tian et al. (2015) investigated the impact of 28 factors along eight dimensions on user ratings. They observed that the size of an app is the most influential factor on user ratings. Bavota et al. (2015) studied the impact of API usage on user ratings, and showed that apps using less fault- and change-prone APIs tend to receive higher user ratings.

A few studies focus on cross-platform apps. Man et al. (2016) proposed a framework to analyze app issues across different mobile app distribution platforms. Hu et al. (2016, 2018) studied the consistency of star ratings of the top free cross-platform apps. They observed that 79% of the studied cross-platform apps do not receive consistent ratings across platforms, and hybrid apps do not guarantee consistent ratings across platforms as well.

Several studies examine the developer response of reviews. McIlroy et al. (2017) studied the value of developers responding to mobile app reviews, and observed that there are positive effects to responding to reviews, with a median increase of 20% in the rating. Hassan et al. (2017b) studied the dialogue between users and developers, and observed that developers and users use the response mechanism as a rudimentary user support tool.

Mining mobile app updates. Several studies focus on mining the updates of mobile apps. Martin et al. (2016) implemented a tool CIRA for Causal Impact Release Analysis on Google Play apps. Their approach reveals important characteristics in a update that may positively affect app rating. McIlroy et al. (2016a) studied the frequency of updates of 10,713 mobile apps and showed that users are not annoyed by the high update frequency, instead they tend to highly rank frequently-updated apps. Hassan et al. (2017a) analyzed the characteristics of emergency updates in the Google Play Store. They identified eight patterns of emergency updates and documented their causes and impact on the user experience.

Mining miscellaneous artifacts. Liu et al. (2017b) presented a method to automatically extract domain knowledge from an app's description. Liu et al. (2017a) mined user behavioural data from Wandoujia, a leading Android app-store service in China, and suggested new open opportunities for the research community. van Angeren et al. (2016) studied the relationships among app developers, and suggested that different app stores should select different governance mechanisms for entry barriers and partnership models. Wang et al. (2018) investigated the reasons behind the removal of apps in the Google Play Store.

Beyond handheld apps. Mujahid et al. (2018) mined user reviews for Android Wear apps, and observed that wearable apps have unique user complaints in comparison to handheld apps.

Concerns regarding studies that mine data from mobile app stores. Recent studies express concerns regarding prior studies that mine mobile app stores' data. Martin et al. (2015) discussed the effects of sampling bias in app store mining and proposed potential ameliorations. Nayebi et al. (2018) observed that tweets about mobile apps are more critical and objective than reviews from the app store, and suggested that other information sources should be considered in future work.

3.2.2 Mining Online Distribution Platforms for Games

Mining data from online distribution platforms for games is an area that is largely unexplored. Sifa et al. (2014) studied the time that users spent on games for 6 million users on the Steam platform. They summarized the Playtime Principle: there are fundamental properties governing playtime, irrespective of the underlying specific properties of the game in question. Cheung et al. (2014) analyzed over 200 game reviews from Amazon and several game review sites, and showed that the first sustained play session is important for players' engagement. Poretski and Arazy (2017) analyzed 45 games and their "modding" (user modification) communities on the game modification distribution platform *NexusMod*. They showed that an active modding community can increase the sales of the modded game.

Several empirical studies examined the social network of the Steam Community. Blackburn et al. (2014) studied cheaters in the Steam Community. They showed that the social network of a player (e.g., whether a player has cheating friends) plays an important role in whether a player becomes a cheater. Becker et al. (2012) analyzed the evolution of the Steam Community social network and examined user groups in the Steam community. Sifa et al. (2015) studied the cross-game behaviour of players in the Steam Community. They analyzed how players that play multiple games on Steam divide their playtime and which games are played by them.

Summary: Extensive research has been conducted in the field of mining mobile app stores, especially for mobile app reviews. The valuable insights from mining mobile app stores lead us to expect a promising outcome of mining online distribution platforms for games. However, little work has been done on mining online distribution platforms for games.

3.3 Software Engineering and Games

Several studies have examined various software engineering aspects of game development. Lewis and Whitehead (2011b) highlighted the potential research topics at the intersection of games and software engineering. Scacchi and Cooper (2015) surveyed the software engineering literature in computer games, and examined the challenges in requirements engineering, software design, and software testing for game software.

Software engineering for traditional non-game v.s. game software. Several studies compare the differences between software engineering for traditional non-game software and game software. Pascarella et al. (2018b) compared game development with traditional software development within the scope of open source systems. They confirmed the significant differences between the development of game and non-game software. Ampatzoglou and Stamelos (2010) examined how software engineering practices are used in game development. They showed that game developers adjust traditional software engineering methods to make them better suited for game development. Murphy-Hill et al. (2014) performed a study with 14 interviewees and 364 survey respondents to elicite substantial differences between video game development and traditional non-game software development practices. Murphy-Hill et al. observed that game developers are hesitant to use automated testing because these tests limit the creativity of game designers. Graham and Roberts (2006) discussed the differences between game development and traditional non-game development, and proposed quality attributes of 3D video games and six tactics to address these quality attributes. Kasurinen et al. (2017) conducted a survey of gaming industry organizations to identify areas that warrant future studies of gaming industry practices. They observed that organizations in the gaming industry either apply mostly agile process models or nothing at all, and that software reusability was a minor concern.

Software development practices for games. Several studies focus on the software development practices for games. Politowski et al. (2016) extracted the development processes of 20 games, and observed that at least 65% of the projects used iterative processes, while the waterfall process is used in at least 30% of the studied projects. Daneva (2017) explored requirement engineering for games by interviewing 16 practitioners and reviewing gameplay requirements documents. They revealed a process of balancing different perspectives of gameplay requirements. Koleva et al. (2015) conducted a survey with game development companies to understand the usage of resources and workflow of coordination during game development, and identified a number of challenges. Scacchi (2011) explored game modding as a method for developing games by extending existing games. Alatalo et al. (2013) proposed a metric to quantify the complexity of using an API to develop networked multiplayer games.

Some studies focus on the quality assurance process of game development. Köhler et al. (2012) investigated how the fidelity of graphic assets in the prototype phase of a game influence the feedback from usability testing. Bécares et al. (2017) presented an approach to automate the beta testing process of games. Varvaressos et al. (2017) developed a runtime verification technique to automatically detect bugs when games are played, to speed up the testing phase. Lewis and Whitehead (2011a) presented a system named Mayet to repair games as they execute. Shafiei and van Breugel (2013) uses the model checker Java PathFinder to catch bugs such as uncaught exceptions for Java games. Ahmed et al. (2017) examined the quality of 55 open source games, and stated that open source games have an acceptable level of correctness, reliability and maintainability.

A few prior studies investigated model-driven game engineering. Guana et al. (2015) presented their experience of using model-driven engineering to design and construct a game engine. Reyno and Carsí Cubel (2009) proposed an approach to automatically prototype 2D platform game using model-driven engineering. Hastjarjanto et al. (2013) introduced a domain-specific language to describe the AI of real-time video games.

Data mining for game engineering. Several prior studies apply data mining to game-related data. Zimmermann et al. (2012) described the practices of analyzing automatically collected game data in Microsoft. Chambers et al. (2005) analyzed two years of game-related data, such as server traffic and player numbers. They demonstrated the difficulty of providing enough resources at launch time of a game and showed that gamers are extremely difficult to satisfy. Wallner (2015) proposed the use of lag sequential analysis (LSA) for analyzing sequential behavior patterns of players. Harpstead et al. (2015) applied log mining to create engagement profiles of game players. Huang et al. (2013) analyzed gameplay data for *Halo Reach*, a popular Xbox game, to investigate what differentiates the best players (players with the highest TrueSkill ratings, a Bayesian scoring system similar to the Elo rating in chess) from regular players. Gagné et al. (2012) described their practice of instrumenting, collecting, and analyzing telemetry data for a real-time strategy game. Shores et al. (2014) studied the game records and user feedback of the *League of Legends* game, and suggested methods to detect toxic players in the game. Drescher et al. (2018) linked in-game data with Twitter activity to analyze the fluctuations in in-game activity of the *Density* game. Linehan et al. (2014) analyzed the play-through videos of four puzzle games to understand how to best pace challenges in games.

Wallner and Kriglstein (2013) surveyed literature on gameplay data visualization, and proposed a categorization of visualization techniques. Medler et al. (2011) and Wallner and Kriglstein (2012) developed tools for visual analytics of gameplay data.

Game architecture. Some studies focus on the architecture of games. Albassam and Gomaa (2013) designed a variable component-based Software Product Line (SPL) architecture for a multi-platform video game. Ollsson et al. (2015) investigated the implementation of the Model-View-Controller (MVC) architecture pattern in 5 games and observed that architectural refactoring increases their software quality. Dragert et al. (2011, 2012) developed reusable AI components to enable the fast development of new AIs in games.

Lessons from game development. Several prior studies focus on the associated challenges with game development. Washburn Jr et al. (2016) studied 155 postmortem retrospectives from game development in which game developers discuss what went wrong and what went right during the development of a game. Washburn et al. showed that planning at the early stage of game development is important. Lewis et al. (2010) presented a taxonomy of 11 types of failures in video games by surveying game failure videos on YouTube.

Petrillo et al. (2009) surveyed the problems in game development and compared them with well-known problems in the traditional non-game software development. Hall (2011) described the software engineering challenges that they faced when designing multi-player outdoor mobile games. Graham (2010) introduced five grand challenges for developing networked games. Smith et al. (2012) defined procedural content generation-based (PCG-based) game design and described the challenges of creating a PCG-based game.

Summary: Prior studies confirm the large difference between developing a traditional non-game and a game software system, raising threats to directly applying knowledges from prior software engineering research to game engineering. Several prior studies investigated various practices of game engineering, or mined gameplay data for individual games. However, there is no prior work that mines online distribution platforms for games to provide valuable insights and suggestions for game development.

CHAPTER 4

Studying Urgent Updates of Popular Games on the Steam Platform

An urgent update is a software update that fixes problems that are deemed critical enough to not be left unfixed until a regular-cycle update. Urgent updates are made in a state of emergency and outside the regular development and test timelines which causes unnecessary stress on the development team. Hence, avoiding the need for an urgent update is important for game developers. We conducted an empirical study of the urgent updates of the 50 most popular games from the Steam platform. As urgent updates are reflections of mistakes in the development and test processes, a better understanding of urgent updates can in turn stimulate the improvement of these processes, and eventually save resources for game developers. In this chapter, we argue that the update strategy that is chosen by a game developer affects the number of urgent updates that are released. Although the choice of update strategy does not appear to have an impact on the percentage of updates that are released faster than the regular cycle or self-admitted hotfixes, games that use a frequent update strategy tend to have a higher proportion of 0-day updates than games that use a traditional update strategy.

An earlier version of this chapter is published in the Empirical Software Engineering Journal (EMSE) (Lin et al., 2017).

4.1 Introduction

In many cases, developers advertise the update notes of new updates of their games through online gaming communities to reach the game players. As such, these update notes offer valuable insights into the update practices of game developers. In particular, we can infer the update cycle of a game, which in turn allows us to identify *urgent updates*.

Urgent updates are deemed critical enough to not be left unreleased until an upcoming regular-cycle update. As urgent updates are usually released in a state of emergency, i.e., to quickly respond to critical errors that are introduced by a previous game update, urgent updates cause unnecessary stress on developers. The stress of these so-called "fire-fighting conditions" can not only lead to inefficient problem solving, but also introduces changes that can easily create new problems (Bohn, 2000), and hence such updates should be avoided by game developers.

In this chapter, we performed an empirical study on urgent updates of the 50 most popular games on the Steam platform. Our goal is to help game developers understand the causes behind urgent updates, and in turn stimulate the improvement of the development and test processes of games. First, we studied the update frequency, update consistency and update strategy of the studied games in a preliminary study. Our preliminary study showed that while 32% of the games follow a frequent update strategy, 68% of the studied games follow a build-up candidate update strategy. Games that follow a build-up candidate update strategy hold off their updates until they release a major update which contains many minor updates. Then, we examined the following questions:

How often do developers release urgent updates? We consider 0-day updates, updates that are released faster than the regular cycle and self-admitted hotfixes to be urgent updates. 80% of the studied games have urgent updates. Games that use a frequent update strategy have a higher proportion of 0-day updates than games that use a build-up candidate update strategy. 46% of the studied games have self-admitted hotfixes.

Why do developers release urgent updates? 36% of the urgent updates are released to make changes to the rules of a game. Feature malfunctions, crashing games and visual bugs are the most commonly-given reasons for releasing urgent updates.

Prior work on urgent updates focus on urgent updates that are released to patch security vulnerabilities in software (Arora et al., 2010; Kim et al., 2011). In addition, Hassan et al. (2017a) studied urgent updates for mobile apps. We are the first, to the best of our knowledge, to empirically study the interesting phenomenon of urgent updates for games.

4.2 Background on the Study of Urgent Updates

In this section, we give background information for our study. First, we briefly describe the release strategies that we study. Then, we discuss related work.

4.2.1 Update Strategies

In this chapter, we classified each studied game into one of two classes, based on the update strategy that the game uses. The first class contains games that follow a traditional update strategy, i.e., these games hold off their updates until they release a major update which contains many minor updates. We call this strategy the *build-up candidate* strategy, to emphasize that the developer "builds up" a release candidate. A characteristic of the build-up candidate strategy is that the number of days between updates is often large (in the order of months or even years).

The second class contains games that release updates frequently. These games release an update as soon as a feature or fix is finished. Hence, the update timeline of these games is filled with minor updates. The characteristic of the frequent update strategy is that the number of days between updates is often small (in the order of days or weeks).

For both update strategies, the number of days between updates may increase as the game matures, for several reasons. For example, a developer may focus on developing new products, while updating older products only when absolutely necessary. Because the number of days between updates may increase over time, we cannot simply classify the games based on the number of days between updates only. In Section 4.4, we discuss our classification of games based on their update strategy.

4.2.2 Related Work

In the remainder of this section, we discussed prior research that is related to our work.

Empirical Studies on Urgent Updates

The majority of empirical studies on urgent updates focus on so-called patch updates, which are updates for security vulnerabilities (Arora et al., 2010; Kim et al., 2011). Arora et al. (2010) showed that the release time of a security hotfix is heavily impacted by how fast a competitor that suffers from the same vulnerability addresses the issue. As the release of the hotfix of the competitor also discloses the vulnerability, it becomes essential for others to fix that vulnerability as well.

Arora et al. (2006) showed that releasing software faster than a competitor can lead to financial benefit despite the high cost of hotfixes.

Kerzazi and Adams (2016) studied 345 releases of a large e-commerce web app and identified 17 recurrent root causes of botched releases, classified into four major categories. Hassan et al. (2017a) studied 1,000 emergency updates of over 10,000 mobile apps in the Google Play Store. Hassan et al. identified 8 patterns of emergency updates and categorize along two dimensions “Updates due to deployment issues” and “Updates due to source code changes”. Hassan et al. suggested that app developers should carefully avoid these patterns.

Our work is the first, to the best of our knowledge, that conducts an empirical study of urgent updates of games.

Empirical Studies on Update Strategies

Prior work has studied the release strategies of various types of software, for example, mobile apps (Nayebi et al., 2016; McIlroy et al., 2016a). Mobile apps are distributed through mobile app stores, which are similar to digital game distribution platforms, as mobile app stores allow users to download, update and comment on mobile apps in one centralized location. Nayebi et al. (2016) show that while mobile app developers mostly prefer frequently releasing updates for an app, users of the app have mixed feelings about frequent updates. As a result, only half of the users automatically install new updates. It is an interesting question whether game users share the same mixed feelings about frequent updates. However, installing a game update is mandatory in Steam, hence these mixed feelings are hard to verify. Nevertheless, having to frequently wait for an update to download and install before one can play a game is likely to frustrate gamers.

McIlroy et al. (2016a) showed that 45% of the updates of frequently-updated mobile apps (i.e., at least bi-weekly) do not provide a rationale for updating. In addition, McIlroy et al. showed that only 1% of the apps is updated at least once a week. In our study, we observe that a large portion (44%) of the games are updated frequently, i.e., often within a week. The most important reason for frequent updates in mobile apps is to fix a bug, which is a consistent observation with our observations about urgent updates for games.

Mäntylä et al. (2013) conducted a case study on Mozilla Firefox about the changes in software testing effort after moving to a rapid release strategy (i.e., releases every six weeks). Mantyla et al. stated that rapid releases lead to a narrower development scope, which allows deeper testing of features and regressions with the highest risk. In addition, the required number of specialized testers grows, in order to sustain testing effort in the rapid release model. Mäntylä et al. concluded that the rapid release strategy does not have a significant impact on the product quality. Souza et al. (2015) studied how transitioning to a rapid release strategy changed the backout rate for Mozilla Firefox. The backout rate describes the rate of patches that are reverted after their release.

Souza et al. observed that the overall backout rate increased under rapid releases but that this increased rate has no effect on users' perception of product quality. da Costa et al. (2016) conducted an empirical study of the impact of Mozilla Firefox switching to a rapid release strategy on the integration delay of addressed issues. da Costa et al. showed that a rapid release strategy may not be able to deliver addressed issues to users faster than through a traditional release strategy. Khomh et al. (2012) empirically studied the development process of Mozilla Firefox during its transition to a rapid release cycle. Khomh et al. observed that although with shorter release cycles, users do not experience significantly more post-release bugs and the bugs are fixed faster, users experience these bugs earlier during software execution. Khomh et al. later extended their work (Khomh et al., 2015) and noted that one of the major challenges when switching to rapid releases is the automation of the release engineering process.

We are the first to study update strategies for games and in particular we examine how the frequency of releasing updates affects the number of urgent updates.

4.3 Our Methodology for Studying Urgent Updates

In this section, we introduce the methodology of our empirical study of urgent updates of popular games. We detail how we selected our subject systems and extracted the needed data to conduct our study. Figure 4.1 gives an overview of our methodology.

4.3.1 Selecting Subject Systems

We selected the 50 most popular games on Steam on January 12, 2016 for the study. The list of top 50 games was provided by Steam Charts (Gray, 2016), a website that

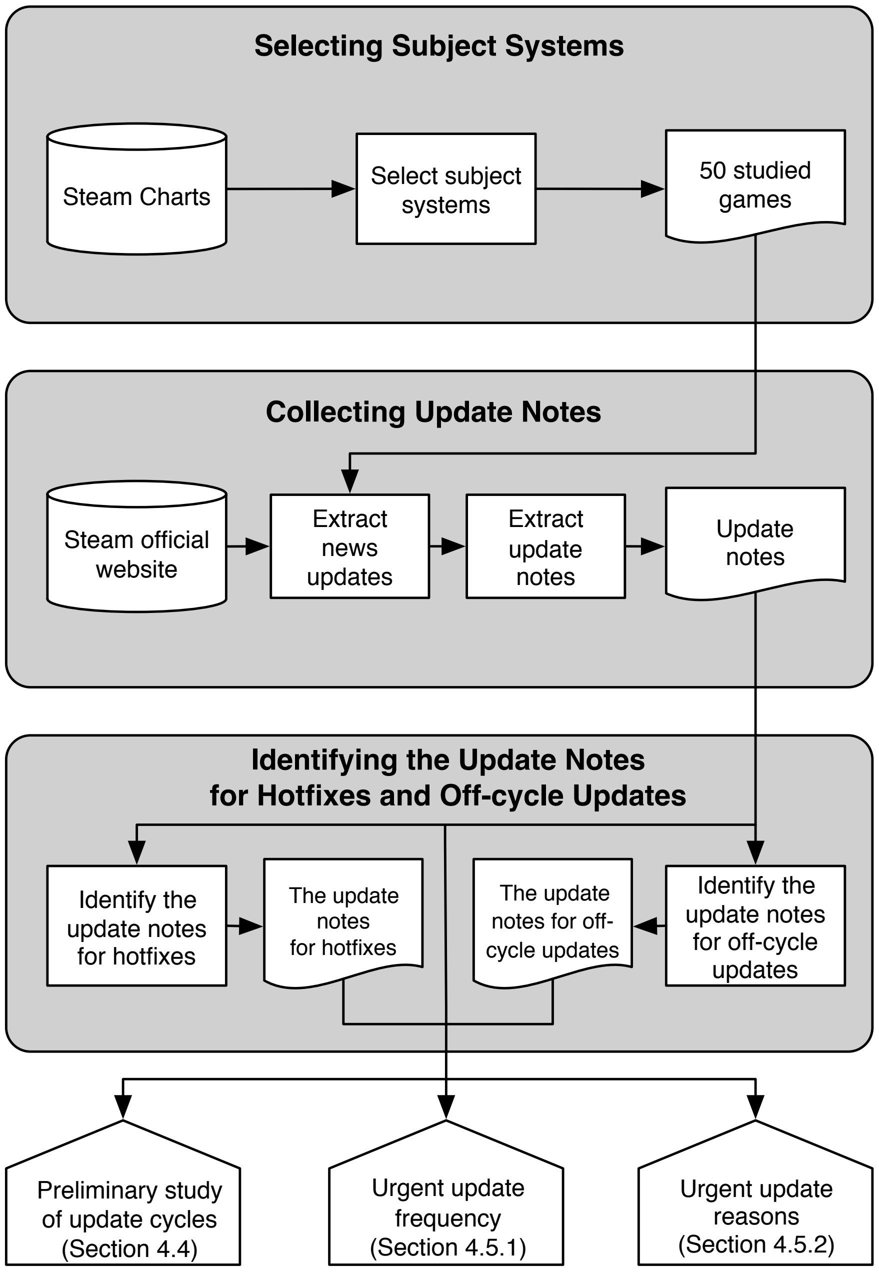
Selecting Subject Systems
Steam Charts
Select subject systems
50 studied games
Collecting Update Notes
Steam official website
Extract news updates
Extract update notes
Update notes
Identifying the Update Notes for Hotfixes and Off-cycle Updates
Identify the update notes for hotfixes
The update notes for hotfixes
The update notes for off-cycle updates
Identify the update notes for off-cycle updates
Preliminary study of update cycles (Section 4.4)
Urgent update frequency (Section 4.5.1)
Urgent update reasons (Section 4.5.2)

Figure 4.1: Overview of our study of urgent updates

ranks games by the number of players on that day. Table 4.1 shows details about the 50 games that we selected for our study.

4.3.2 Collecting Update Notes

We used the update notes that are posted on the channels in the Steam Community to infer the update cycles of each studied game. As mentioned in Chapter 2, developers do not necessarily need to announce all updates that they make. We used the published update notes to get a lower bound of the number of updates for each of the studied games.

Although the Steam Community has a special channel available for update notes, called the *Product Updates* channel, we observed that many update notes are not posted on that channel but on other channels instead (e.g., the *Community Announcements* channel). To avoid missing any update notes, we extracted all information across all news channels for all studied games. The "Related news" page of a game in the Steam Store[1] aggregates all news updates that are related to that game from all available Steam Community channels. These news updates include for example game announcements, promotions and update notes. Table 4.2 shows an example of an update note for the *Team Fortress 2* game.

We extracted all 11,970 news updates for the studied games using a custom-written crawler. We performed the following steps to extract update notes from the news updates:

1. We kept all news updates that are posted on the *Product Release* or *Product Update* channel.

[1]E.g., related news for Dota 2: `http://store.steampowered.com/news/?appids=570`

Table 4.1: Basic information about the studied games on Steam, sorted by the number of players (as of January 12, 2016)

Title	**Developer**	**Genre**	**Release year**	**# of players**	**Early access**[2]
Dota 2	Valve	Strategy	2013	858890	
Counter-Strike: Global Offensive	Valve	Action	2012	563938	
Football Manager 2016	SPORTS INTERACTIVE	Sports	2015	68949	
Fallout 4	Bethesda Game Studios	RPG	2015	61214	
Grand Theft Auto V	Rockstar North	Adventure	2015	56419	
Team Fortress 2	Valve	Action	2007	56390	
ARK: Survival Evolved	Studio Wildcard	RPG	2015	50522	✓
Sid Meier's Civilization V	Firaxis Games	Strategy	2010	45352	
Garry's Mod	Facepunch Studios	Simulation	2006	39694	
The Elder Scrolls V: Skyrim	Bethesda Game Studios	RPG	2011	36107	
Warframe	Digital Extremes	Action	2013	35983	
Rust	Facepunch Studios	RPG	2013	35128	✓
Rocket League	Psyonix	Sports	2015	34342	
Arma 3	Bohemia Interactive	Strategy	2013	32294	
Counter-Strike	Valve	Action	2000	26814	
H1Z1 : Just Survive	Daybreak Game Company	Adventure	2015	24577	✓
Euro Truck Simulator 2	SCS Software	Simulation	2013	21689	
Call of Duty: Black Ops III	Treyarch	Adventure	2015	21643	
Terraria	Re-Logic	RPG	2011	20594	
Unturned	Smartly Dressed Games	Casual	2014	20466	✓
PAYDAY 2	OVERKILL - a Starbreeze Studio.	RPG	2013	17064	
SMITE	Hi-Rez Studios	Action	2015	16510	
The Witcher 3: Wild Hunt	CD PROJEKT RED	RPG	2015	14415	
War Thunder	Gaijin Entertainment	Simulation	2013	14364	
Path of Exile	Grinding Gear Games	RPG	2013	14159	
Left 4 Dead 2	Valve	Action	2009	13866	
Europa Universalis IV	Paradox Development Studio	Strategy	2013	13112	
Counter-Strike: Source	Valve	Action	2004	13068	
Tom Clancy's Rainbow Six Siege	Ubisoft Montreal	Action	2015	12742	
DayZ	Bohemia Interactive	Action	2013	11505	✓
Total War: ROME II - Emperor Edition[1]	Creative Assembly	Strategy	2013	10795	
Trove	Trion Worlds	RPG	2015	10216	
Mount & Blade: Warband	TaleWorlds Entertainment	RPG	2010	9976	
Don't Starve Together	Klei Entertainment	Simulation	2014	9876	✓
Borderlands 2	Gearbox Software	RPG	2012	9720	
METAL GEAR SOLID V: THE PHANTOM PAIN[1]	Konami Digital Entertainment	Adventure	2015	9513	
XCOM: Enemy Unknown	Firaxis Games	Strategy	2012	9253	
Age of Empires II HD	Skybox Labs	Strategy	2013	8523	
7 Days to Die	The Fun Pimps	Simulation	2013	8253	✓
Cities: Skylines	Colossal Order Ltd.	Strategy	2015	7369	
Company of Heroes 2	Relic Entertainment	Strategy	2013	7074	
Arma 2: Operation Arrowhead	Bohemia Interactive	Strategy	2010	7023	
AdVenture Capitalist	Hyper Hippo Games	Casual	2015	6946	
Total War: ATTILA	Creative Assembly	Strategy	2015	6843	
Hurtworld	Bankroll Studios	Simulation	2015	6806	✓
Undertale	tobyfox	RPG	2015	6748	
Brawlhalla	Blue Mammoth Games	Action	2015	6530	✓
Just Cause 3	Avalanche Studios	Adventure	2015	6518	
Dying Light: The Following - Enhanced Edition[1]	Techland	RPG	2015	6360	
DARK SOULS II: Scholar of the First Sin[1]	FromSoftware, Inc	RPG	2015	6342	

1. We will use shortened game names throughout the rest of the chapter for brevity in the tables.
2. Early access games allow customers to purchase the game during its public beta period while developers continue working on the game.

Table 4.2: Update note for the *Team Fortress 2* game

Title	Team Fortress 2 Update Released
Channel	Product Updates
Date	12 Oct, 2015
An update to Team Fortress 2 has been released. The update will be applied automatically when you restart Team Fortress 2. The major changes include: - Fixed a client crash related to the contract menu. - Fixed an issue where some players could not use some of the crafting recipes - Running in textmode now places the client in insecure mode - Updated the localization files	

2. For the remainder of the news updates, we removed all news updates that are posted on the Steam client announcements channel, or channels that are related to game reviews, or channels that are known to contain only crossposts.

3. We removed all news updates of which the title does not contain the words *update, release, patch, hotfix, change log* **OR** a version number.

4. The news updates that are left, together with the news updates from step 1 are considered as update notes.

We must perform step 2 because posts in these channels can contain a review of another update, which will negatively affect the precision of step 3. We manually identified the following channels that are related to game reviews and the Steam client: *Rock. Paper. Shotgun, PC Gamer, Shacknews, Kotaku, Eurogamer, Announcements, Steam Blog, Press Releases, Client Updates.* In addition, we manually identified the following channels that contain only crossposts: *TF2 Official Blog, Left 4 Dead Official Blog, Portal 2 Official Blog.* As these channels are for games developed by Valve, i.e., the developer of Steam, update notes are posted to the *Product Update* channel as well. We removed all news updates that are posted on irrelevant channels.

We identified 2,672 update notes for the 50 studied games. In order to validate the precision and recall of our extraction steps, we manually analyzed a statistically-representative random sample of 372 news updates (95% confidence level and 5% confidence interval, taken from the 11,970 news updates of the studied games) and count the news updates that do not contain update notes. The manual analysis of the representative sample shows that our extraction steps have a precision of 88% and a recall of 87%. In order to further enhance the precision of our data, we manually checked the identified update notes and removed 253 news updates that do not contain update notes, leaving 2,419 update notes for our study.

4.3.3 Identifying the Update Notes for Hotfixes and Off-cycle Updates

We considered three types of irregular updates as urgent updates in this study:

1. **Self-admitted hotfixes:** Game updates that are described by developers as hotfixes.

2. **Off-cycle updates:** Game updates that are released outside the regular update cycle of a game.

3. **0-day updates:** updates that are released on the same day.

We identified update notes for self-admitted hotfixes using the regular expression **(hot.?fix)**[2] on the titles and contents of update notes. Using this regular expression, we identified 163 update notes for self-admitted hotfixes. We manually checked all

[2]We attempted to extend this regular expression with more terms such as ‘patch’ and ‘emergency’, however, we observed that these terms incorrectly match too many update notes that are not for hotfixes.

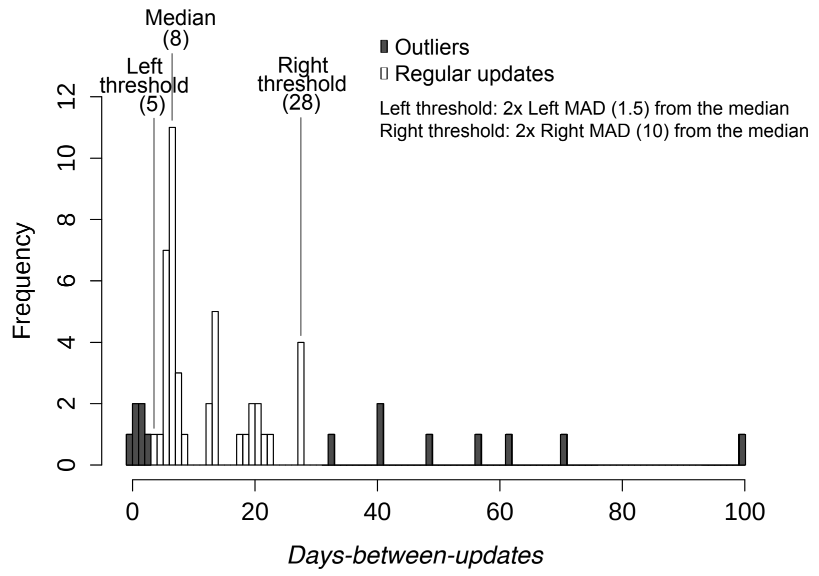

Figure 4.2: An example of detecting outliers for the *Warframe* game.

of them and exclude 15 wrongly identified update notes, leaving 148 update notes for self-admitted hotfixes. The wrongly identified update notes are regular update notes that contain a statement such as *“We will keep monitoring feedbacks and push hotfixes if necessary"*.

To identify off-cycle updates, we calculated the *days-between-updates* for all adjacent updates for all games. We then used the Median Absolute Deviation (MAD) to identify the outliers of the *days-between-updates*, i.e., updates that take a statistically significantly longer or shorter period than is usual for that game. The MAD is a robust statistic which measures the variability of a univariate sample of quantitative data. The MAD is defined as the median of the absolute deviations from the data’s median. We used the MAD to identify outliers as suggested by Leys et al. (2013), who show that using the absolute deviation around the median outperforms using the standard deviation around the mean when detecting outliers. Generally, if a value is a certain number

Table 4.3: Dataset description of our study of urgent updates

# of studied games	50
# of news updates	11,970
# of update notes for:	
All game updates	2,419
Self-admitted hotfixes	148
Off-cycle updates	411
0-day updates	162

of MAD away from the median of the residuals, that value is classified as an outlier. However, Figure 4.4 shows that the distributions of *days-between-updates* are highly unsymmetric. We addressed this problem by using the *Double MAD* as suggested by Rosenmai (2013), i.e., we calculated the MAD for the left and right side of the median of the distribution, and used the left MAD to identify outliers on the left tail, while using the right MAD to identify outliers on the right tail. Miller (1991) proposes that depending on the stringency of the researcher's criteria, the threshold for the number of MADs can be 3 (very conservative), 2.5 (moderately conservative) or 2 (poorly conservative). After a preliminary experiment on the *days-between-updates* in our dataset, we selected 2 as the threshold for our dataset. Figure 4.2 shows an example of detecting outliers for the *Warframe* game using the *Double MAD*. We identified 411 off-cycle updates in total.

4.3.4 Urgent Updates Dataset Description

Table 4.3 presents the description of our collected dataset.

4.4 Preliminary Study of the Update Cycles of the Studied Steam Games

In this section, we present our preliminary study of the update cycles of the studied games. The goal of the preliminary study is to get a better understanding of the update cycle of the studied games by identifying their update frequency, update cycle consistency and update strategy. First, we explain our approach, then we present the findings of our preliminary study.

Approach: Because developers are not obliged to publish update notes for a game update, nor does Steam provide an exhaustive list of game updates, we used the published update notes to get a lower bound of the number of updates for each of the studied games.

To study update frequency, we first removed games with less than 3 updates, as such games do not provide enough information for us to infer their update cycle. We calculated the median and mode of the *days-between-updates* (i.e., the *days-between-updates* that occur most often) of all the studied games as metrics for update frequency.

To study update cycle consistency, we calculated Fisher's kurtosis (Zwillinger and Kokoska, 1999) of the *days-between-updates*. Kurtosis expresses the peakedness of a distribution. The normal distribution has a kurtosis of 3, and a kurtosis higher than 3 indicates that the distribution has a higher peak than the normal distribution. A higher kurtosis of the *days-between-updates* indicates that the game has a more consistent update cycle, as the *days-between-updates* are then centered around a single value.

Table 4.4 shows the update frequency and update cycle consistency metrics for all studied games. We used these metrics to manually classify all the games into two

Table 4.4: Updates of studied games on Steam, sorted by the kurtosis of *days-between-updates* (as of January 12, 2016)

				days-between-updates[1]		
Title	**Updates**	**% Self.adm hotfixes**	**% Off-cycle updates**[2]	**Median**	**Mode**[3]	**Kurtosis**
Team Fortress 2	464	0	13	3.0	1(100)	82.51
Don't Starve Together	91	43	15	3.0	1(25)	55.27
Unturned	158	1	20	2.0	1(64)	46.41
Counter-Strike: Source	84	0	21	7.0	0(7)	43.04
Left 4 Dead 2	134	0	21	7.0	7(23)	30.83
Borderlands 2	32	3	19	19.0	1,2,5,9,27,28(2)	22.09
Counter-Strike	29	0	17	2.0	1(10)	20.46
7 Days to Die	68	28	13	5.0	1(11)	18.35
Company of Heroes 2	26	0	15	13.0	0(8)	17.67
Arma 2: Operation Arrowhead	19	5	11	53.5	16(2)	13.61
Counter-Strike: Global Offensive	90	0	52	7.0	7(29)	13.28
Path of Exile	70	9	13	6.0	3(8)	12.82
DayZ	25	48	4	15.0	0,2,28(3)	12.46
Garry's Mod	66	3	17	13.0	3(8)	11.64
Dota 2	281	0	12	3.0	1(77)	11.12
Brawlhalla	75	0	8	6.0	1(13)	10.29
Dying Light: The Following - E. E.	17	12	12	13.0	4,11(2)	9.48
Euro Truck Simulator 2	33	9	15	21.5	7(3)	9.04
Arma 3	20	5	35	21.0	15,21(2)	8.88
Terraria	12	0	25	13.0	8(2)	8.78
Warframe	58	10	22	8.0	7(11)	8.57
Rust	12	0	17	12.0	6(2)	7.89
Age of Empires II HD	22	5	14	13.0	6,7(2)	7.41
War Thunder	60	2	22	5.0	1(9)	6.78
Trove	42	31	7	4.0	1(9)	6.66
PAYDAY 2	148	19	19	3.0	1(39)	6.53
Sid Meier's Civilization V	37	14	38	22.0	22(3)	5.76
Mount Blade: Warband	29	10	14	25.5	3,12(2)	5.40
AdVenture Capitalist	9	0	22	24.0	37(2)	5.16
The Witcher 3: Wild Hunt	17	6	24	4.0	0(6)	5.08
Just Cause 3	7	0	14	9.0	3(2)	3.97
Call of Duty: Black Ops III	8	0	13	5.0	0(2)	3.51
Europa Universalis IV	23	35	17	10.5	0(4)	3.47
H1Z1 : Just Survive	61	10	18	5.5	7(9)	2.92
XCOM: Enemy Unknown	10	0	20	34.0	-[4]	2.71
The Elder Scrolls V: Skyrim	10	0	10	49.0	-[4]	2.64
Total War: ROME II - E. E.	15	0	13	13.5	7(2)	2.50
Rocket League	13	8	8	13.0	20(2)	2.36
Hurtworld	5	0	0	5.5	-[4]	2.28
Total War: ATTILA	6	17	17	48.0	-[4]	1.92
Fallout 4	6	0	17	5.0	9(2)	1.62
Cities: Skylines	6	0	0	11.0	-[4]	1.62
Tom Clancy's Rainbow Six Siege	4	0	0	6.0	-[4]	1.50
Grand Theft Auto V	5	0	0	44.5	-[4]	1.44
ARK: Survival Evolved	6	0	0	20.0	-[4]	1.43
Football Manager 2016	2	0	-	-	-	-
SMITE	2	0	-	-	-	-
METAL GEAR SOLID V: THE P. P.	1	0	-	-	-	-
Undertale[5]	0	-	-	-	-	-
DARK SOULS II: S. of the F. S.	1	0	-	-	-	-

1. The *days-between-updates* metrics are not calculated for games with less than 3 updates.
2. The off-cycle updates are not identified for games with less than 3 updates.
3. Between parentheses we show the numbers of times that the mode occurred. It is possible to have multiple modes with the same number of occurrences.
4. All *days-between-updates* of that game occur once, hence there is no mode.
5. No metrics are calculated for this game because it has no released updates on Steam.

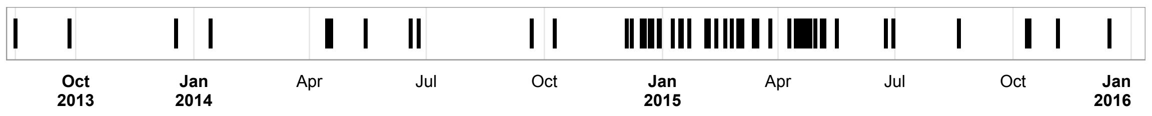

Figure 4.3: Update timeline of the *War Thunder* game. Each vertical line represents an update.

classes: games that follow a frequent update strategy, and games that use a build-up candidate update strategy. For each studied game, we:

1. Examined the median and mode of the *days-between-updates*, and compared those numbers with the total number of updates.

2. Examined the update timeline of the game. Figure 4.3 shows the update timeline of the *War Thunder* game as an example.

3. Examined the update notes when necessary.

4. Classified the game into the frequent update strategy or the build-up candidate update strategy based on the information that is obtained from step 1 to 3.

To verify our classification, two researchers including the author of the thesis and a postdoctoral fellow performed the classification independently, and then compared the results. Only 5 games were classified differently by the two researchers, and the differences were easy to resolve after discussion. There was one game (the *War Thunder* game) which was classified into both classes. Figure 4.3 shows the update timeline of the *War Thunder* game. From Figure 4.3, we can conclude that between December 2014 and June 2015 the game appears to follow a frequent update strategy, while the game follows a build-up candidate update strategy during other time periods. One possible explanation is that the developer was experimenting with the frequent update strategy for half a year and decided to switch back to the build-up candidate update strategy after that. Another explanation is that the developer did not publish update notes for all updates outside the frequent update period. Because we were unable to find the explanation even after a manual study of the update notes, we decided to classify the *War Thunder* game into both update strategies. We did not consider the data for the *War Thunder* game in the rest of our calculations to avoid confusion.

For each studied game, we calculated the percentage of faster off-cycle updates, i.e., off-cycle updates that take less time to release compared to the regular update cycle, and slower off-cycle updates, i.e., off-cycle updates that take more time to release compared to the regular update cycle.

We used the Wilcoxon signed-rank test and Wilcoxon rank sum test to decide whether the distributions of the metrics of update cycles are significantly different. The Wilcoxon signed-rank test is a paired, non-parametric statistical test of which the null hypothesis is that two input distributions are identical, while the Wilcoxon rank sum test is unpaired. If the p-value computed by a test is smaller than 0.05, we conclude that the two input distributions are significantly different. On the other hand, if the p-value is larger than 0.05, the difference between the two input distributions is not significant.

The Wilcoxon tests determine only whether two distributions are different, but not the magnitude of the difference. Therefore, we computed Cliff's delta d (Long et al., 2003) effect size to quantify the difference of the distributions. We used the following threshold for interpreting d, as proposed by Romano et al. (2006):

$$\text{Effect size} = \begin{cases} \textit{negligible}\,(N), & \text{if } |d| \leq 0.147. \\ \textit{small}\,(S), & \text{if } 0.147 < |d| \leq 0.33. \\ \textit{medium}\,(M), & \text{if } 0.33 < |d| \leq 0.474. \\ \textit{large}\,(L), & \text{if } 0.474 < |d| \leq 1. \end{cases}$$

4.4.1 Update Frequency

Many studied games have periods in which they release frequently. Table 4.4 shows that 20 out of 45 (44%) studied games have a median *days-between-updates* that is equal to or less than 7 days, i.e., at least 50% of the updates of these games are released within a week after the previous update. Moreover, in 81% of the studied games, at least one of the modes of the *days-between-updates* is smaller than 7, indicating that these games have periods in which they release frequently.

One possible explanation for the high number of frequent updates is the rich interaction between game developers and players. Games tend to have a more engaged and interactive ecosystem than traditional software or mobile apps through channels such as discussion lists, Twitter, YouTube videos, Twitch.tv, the Steam Community, official websites of games and fan websites. Hence, the gaming community is able to provide feedback to game developers quickly, and game developers tend to address such community feedback in a rapid pace as well.

4.4.2 Update Consistency

Most studied games do not have a consistent update cycle. Figure 4.4 shows the distribution of the *days-between-updates* of the 16 games that have the highest kurtosis

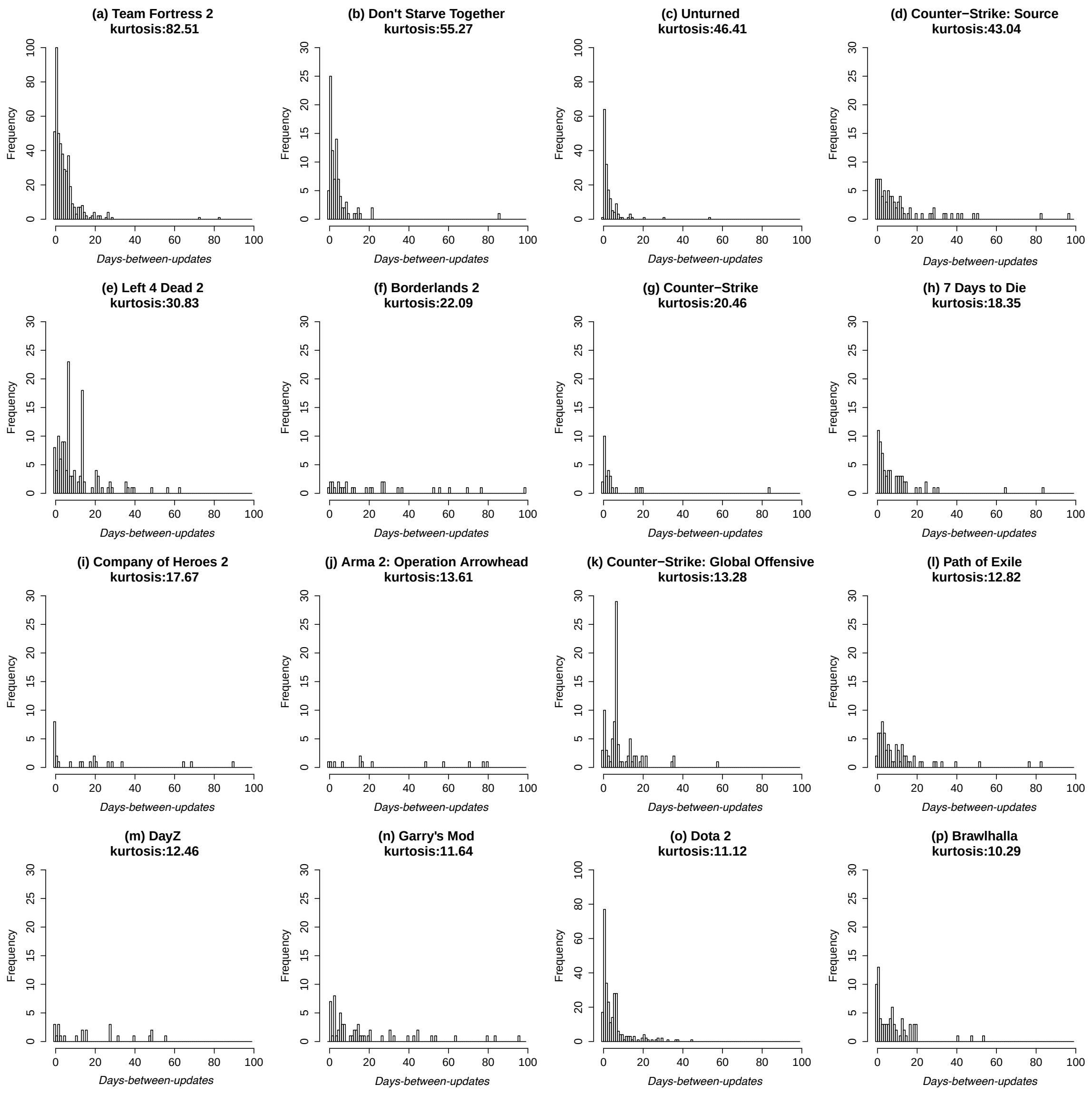

Figure 4.4: Histogram of the *days-between-updates* of the 16 games with the highest kurtosis for that metric. *days-between-updates* greater than 100 are removed to improve the clarity of the figure.

for the *days-between-updates* metric. Table 4.4 shows that only 7 of 45 (16%) of the games have a kurtosis that is higher than 20, and only 16 (36%) of the games have a kurtosis that is higher than 10. Figure 4.4(f) indicates that even for games with a kurtosis that is higher than 20, the update cycle may not be consistent. We look into the *days-between-updates* of the *Borderlands 2* game and find that one update has a *days-between-updates* of 394, while the kurtosis of the *Borderlands 2* game is 22.09. The reason for the high kurtosis despite the large value of *days-between-updates* is that the long tail makes the distribution look more peaked. Hence, kurtosis alone is not enough to describe the consistency of the update cycle of a game.

16% of the games often update on a specific day. Figure 4.4 (e) and (k) hint at an update cycle that is different from most other stable games. The *Left 4 Dead 2* game and the *Counter-Strike: Global Offensive* game have a mode of the *days-between-updates* of 7. In addition, the second-most occurring *days-between-updates* of the *Left 4 Dead 2* game is 14 days. We manually look into these two games and find that most releases of the *Left 4 Dead 2* game are released on Fridays, and most releases of the *Counter-Strike: Global Offensive* game are released on Wednesdays.

Table 4.4 shows that there are 7 of 45 (16%) games for which one of the most occurring values of the *days-between-updates* is 7 days, indicating that 16% of the games often update on a specific day.

4.4.3 Update Strategy

68% of the studied games use a build-up candidate update strategy. Table 4.5 shows the results of our update strategy classification. 68% of the studied games follow the

Table 4.5: Update strategies and off-cycle updates of studied Steam games, sorted by the number of players (as of Jan 12, 2016)

	Update strategy				
Title	**Frequent update**	**Build-up candidate**	**% Faster off -cycle updates**[2]	**% Slower off -cycle updates**[2]	**% 0-day updates**[2]
Dota 2	✓		0	12	6
Counter-Strike: Global Offensive	✓		21	31	3
Football Manager 2016[1]	-	-	-	-	0
Fallout 4		✓	17	0	0
Grand Theft Auto V		✓	0	0	0
Team Fortress 2	✓		0	13	11
ARK: Survival Evolved		✓	0	0	0
Sid Meier's Civilization V		✓	19	19	5
Garry's Mod		✓	0	17	0
The Elder Scrolls V: Skyrim		✓	0	10	0
Warframe		✓	10	12	2
Rust	✓		0	17	8
Rocket League		✓	0	8	0
Arma 3		✓	5	25	0
Counter-Strike	✓		0	17	7
H1Z1 : Just Survive	✓		3	15	3
Euro Truck Simulator 2		✓	0	15	0
Call of Duty: Black Ops III		✓	0	13	25
Terraria		✓	8	17	0
Unturned	✓		0	20	1
PAYDAY 2	✓		0	19	11
SMITE[1]	-	-	-	-	0
The Witcher 3: Wild Hunt		✓	0	24	35
War Thunder	✓	✓	0	22	10
Path of Exile	✓		0	13	3
Left 4 Dead 2	✓		6	15	6
Europa Universalis IV		✓	0	17	17
Counter-Strike: Source	✓		0	21	8
Tom Clancy's Rainbow Six Siege		✓	0	0	0
DayZ		✓	0	4	12
Total War: ROME II - E. E.		✓	0	13	0
Trove	✓		0	7	2
Mount Blade: Warband		✓	0	14	0
Don't Starve Together	✓		0	15	5
Borderlands 2		✓	0	19	3
METAL GEAR SOLID V: THE P. P.[1]	-	-	-	-	0
XCOM: Enemy Unknown		✓	0	20	0
Age of Empires II HD		✓	0	14	5
7 Days to Die	✓		0	13	0
Cities: Skylines		✓	0	0	0
Company of Heroes 2		✓	0	15	31
Arma 2: Operation Arrowhead		✓	0	11	5
AdVenture Capitalist		✓	11	11	0
Total War: ATTILA		✓	0	17	0
Hurtworld		✓	0	0	0
Undertale[1]	-	-	-	-	0
Brawlhalla	✓		0	8	13
Just Cause 3		✓	0	14	0
Dying Light: The Following - E. E.		✓	0	12	6
DARK SOULS II: S. of the F. S.[1]	-	-	-	-	0

1. The metrics are not calculated for games with less than 3 updates.
2. Percentage of all updates.

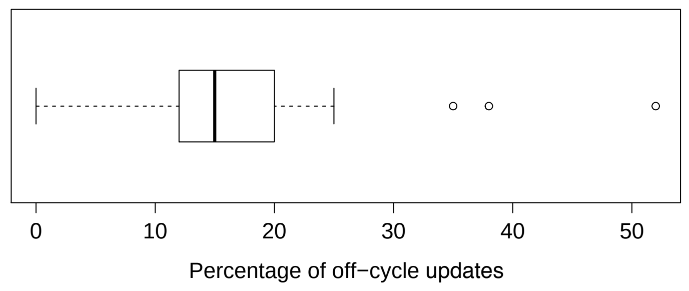

Figure 4.5: Distribution of the percentage of off-cycle updates of studied games. Each dot represents a studied game.

more traditional build-up candidate update strategy. 32% of the games release updates frequently.

Games from the same developers follow the same update strategy. The *Left 4 Dead 2* game and the *Counter-Strike: Global Offensive* game mentioned above are both developed by *Valve*. Table 4.5 shows that all games developed by *Valve* (i.e., the *Team Fortress 2* game, the *Left 4 Dead 2* game, the *Counter-Strike* game, the *Counter-Strike: Source* game, the *Counter-Strike: Global Offensive* game, and the *Dota 2* game) use the frequent update strategy.

In addition, both the *Sid Meier's Civilizationő V* game and the *XCOM: Enemy Unknown* game from *Firaxis Games* use the build-up candidate update strategy. The same phenomenon can be observed from other games that are developed by the same developers (i.e., *Facepunch Studios, Creative Assembly, Bohemia Interactive, Bethesda Game Studios*), suggesting that games from the same developer follow the same update strategy.

The studied games have a median of 15% off-cycle updates. Table 4.4 shows the percentage of off-cycle updates of each studied game. Figure 4.5 shows the distribution of the percentage of off-cycle updates for all the studied games. Although the percentage of off-cycle updates for games varies from 0% to 52%, half of them are between 12%

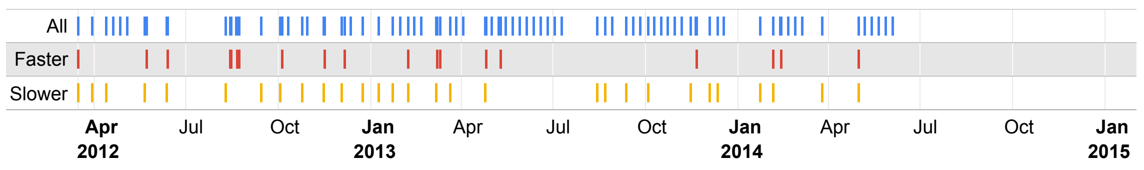

Figure 4.6: Release timeline of the *Counter-Strike: Global Offensive* game. Each vertical line represents an update. There were no updates in 2015 and 2016, hence we omitted these years from the timeline for clarity.

to 20%, with a median of 15% off-cycle updates. The game with the highest percentage (52%) of off-cycle updates is the *Counter-Strike: Global Offensive* game. Figure 4.6 shows the release timeline of the *Counter-Strike: Global Offensive* game. For clarity, we highlighted the faster and slower off-cycle updates on separate timelines. We observed that the update cycle of the *Counter-Strike: Global Offensive* game is fairly consistent. However, there are several periods in which the developers do not release updates. All updates that are released after such an inactive period, are slower off-cycle updates, explaining the relatively large number of slower off-cycle updates that are identified by our approach.

Most off-cycle updates are slower off-cycle updates. Table 4.5 shows the percentage of slower and faster off-cycle updates for each of the studied games. Figure 4.7 shows the distribution of the percentage of slower and faster off-cycle updates for all studied games. All the studied games have at least as many slower off-cycle updates as faster off-cycle updates. The Wilcoxon signed-rank test shows that the difference between the two distributions is significant with a large effect size.

A possible explanation is that games require less updates as they mature. Hence, the *days-between-updates* increases with time, causing these updates to be identified

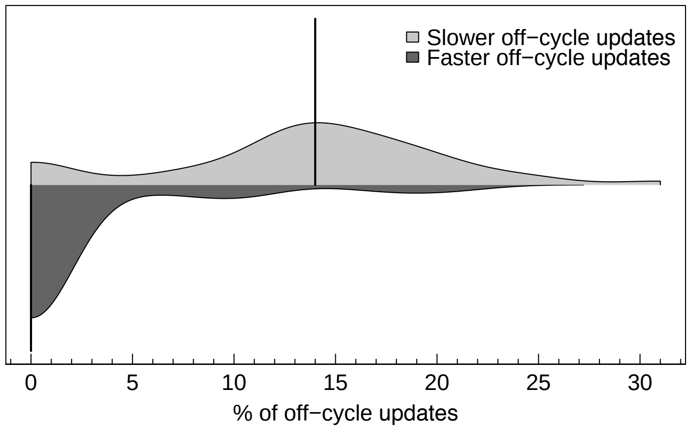

Figure 4.7: Distribution of the percentage of off-cycle updates (of all updates) of each studied game. The vertical lines represent the median. The distributions are significantly different with a large effect size.

as slower off-cycle updates. We studied the faster off-cycle updates in Section 4.5.1 and Section 4.5.2, and we discuss slower off-cycle updates in Section 4.5.4.

There is no difference in the percentage of off-cycle updates or hotfixes between games that follow a frequent update strategy and games that follow a build-up candidate update strategy. Figure 4.8 shows the distribution of the percentage of faster off-cycle updates (of all updates). Figure 4.9 shows the distribution of the percentage of slower off-cycle updates. The Wilcoxon rank sum test shows that the distributions of faster and slower off-cycle updates of the two update strategies are not significantly different. In addition, the Wilcoxon rank sum test also shows that the distributions of hotfixes of the two update strategies are not significantly different, indicating that the choice of update strategy does not appear to have an impact on the percentage of off-cycle updates or hotfixes.

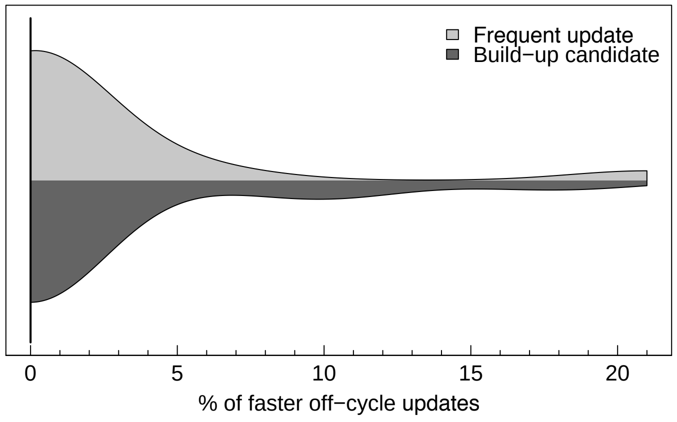

Figure 4.8: Distribution of the percentage of faster off-cycle updates (of all updates) of each studied game. The vertical lines represent the median. The distributions are not significantly different (p > 0.05).

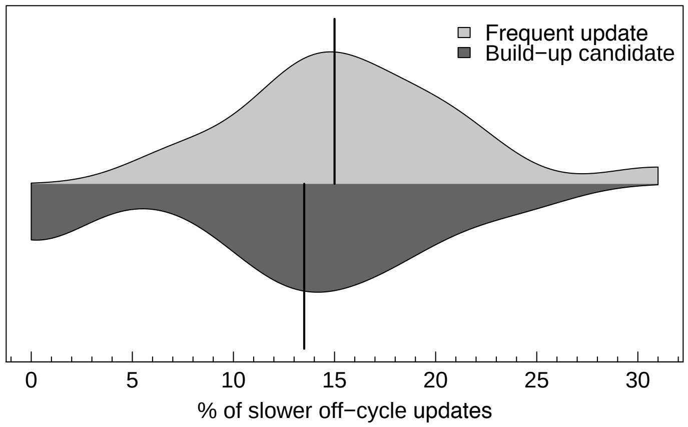

Figure 4.9: Distribution of the percentage of slower off-cycle updates (of all updates) of each studied game. The vertical lines represent the median. The distributions are not significantly different (p > 0.05).

4.5 Results of our Study of Urgent Updates of Popular Steam Games

In this section, we study the urgent updates of popular Steam games. First, we explain the motivation and approach of our empirical study. Finally, we present our findings.

Motivation: Urgent updates are updates that are released to fix an urgent issue that is introduced in the previous botched update. Urgent updates are usually released in a state of emergency and developed outside of the regular update cycle. Therefore, urgent updates tend to be costly (Tassey, 2002), and should be avoided by game developers.

In this study, we consider *0-day updates* (i.e., updates with a *days-between-updates* of 0), off-cycle updates that are released faster than the regular cycle, and self-admitted hotfixes as urgent updates. We study the reasons given in the update notes of urgent updates to get a better understanding of what drives game developers to release urgent updates. With this understanding, game developers can pay more attention to issues that are likely to lead to urgent issues, in order to avoid the need for urgent updates at a later stage.

Approach: First, we studied the frequency of urgent updates. To study frequency, we analyzed the data that we collected as described in Section 4.3. Second, we studied the reasons that are given by developers in their update notes for releasing urgent updates. We manually extracted and categorized the reasons for urgent updates from their update notes. We performed an iterative process that is similar to *Coding* (Seaman et al., 2008; Seaman, 1999) for identifying which reasons lead to urgent updates. The procedure is shown in Listing 4.1.

```
Inputs = All urgent updates, a list of reasons leading to urgent
   updates (which is initially empty)

For each urgent update:
   Manually examine the content of this urgent update.

   If the urgent update matches an existing reason:
      Label the urgent update with that/those reason(s).

   Else:
      Add a new reason to the list of reasons leading to urgent
         updates.
      Restart labelling with new list of reasons.

Outputs = All urgent updates (labelled with appropriate reasons),
   and a list of reasons leading to urgent updates.
```

Listing 4.1: Our coding process for urgent updates

We manually examined the update notes for 162 0-day releases, 47 faster off-cycle updates, and 148 self-admitted hotfixes. We read all release notes and labelled them with one or more reasons for releasing the urgent update. For example, if an urgent update contains a fix for an issue that is related to crashes and performance, we label the urgent update with both the 'CrashingGame' and 'Performance' reasons. Note that we only focused on the changes in the update notes that fix issues rather than those that add features, as the fixes are more likely to help us understand the reasons that drive developers to release urgent updates.

During our analysis, we identified 11 reasons from the update notes of urgent updates. Table 4.6 shows all reasons with their description and an example that is taken from a studied update note. A researcher who collaborated in the study has manually validated the author of the thesis's analysis of reasons that are given in the update notes for urgent updates. The researcher tagged a statistically-representative random

Table 4.6: Identified reasons for releasing urgent updates

Reason	**Description**	**Example**
Functional	Feature malfunctions	*“Fixed save game does not save your minibike"*
CrashingGame	Game crashes	*“Client crashes on some PC’s with intel video card have been fixed."*
RuleLoophole	Loophole in a rule of the game (i.e., a ‘bug’ in a rule)	*“Overlords of colonies and Protectorates can no longer transfer trade power"*
RuleChange	Change of numerical parameter in a rule of the game	*“Lowered dog chase give up time to 18 seconds"*
Content	Fix for an element in the game (e.g. map or weapon)	*“Fixed the Sobek and Torid weapons so that they can be fired when coming out of a sprint"*
Visual	Bug related to visual effects	*“Fixed rain striped effect on surfaces"*
Sound	Bug related to sound effects	*“Flash thunder and weather sounds on entering game fixed"*
UserInteraction	User interaction related bug	*“The ‘End Turn’ button incorrectly displays ‘Please Wait’ rather than ‘Unit Needs Orders’"*
Performance	CPU, network, memory, or disk performance related issues (including online gaming issues such as desynchronization or network lag)	*“Fixed a dedi server taking full CPU time of a single core even if no user was connected"*
Localization	Error related to languages or regions	*“Fixed a localization issue in English"*
Security	Security vulnerability	*“Fixes the steam ID spoofing or account hijacking bug"*

sample of 76 update notes (95% confidence level, 10% confidence interval, out of 357 update notes) with reasons from the set of reasons that were identified by the author of the thesis. Both researchers disagreed on only 5 out of the 76 update notes. All disagreements were for update notes that contained a very game-specific description of the update, which were misinterpreted by the researcher. Hence, after a short discussion, the disagreements were straightforward to resolve.

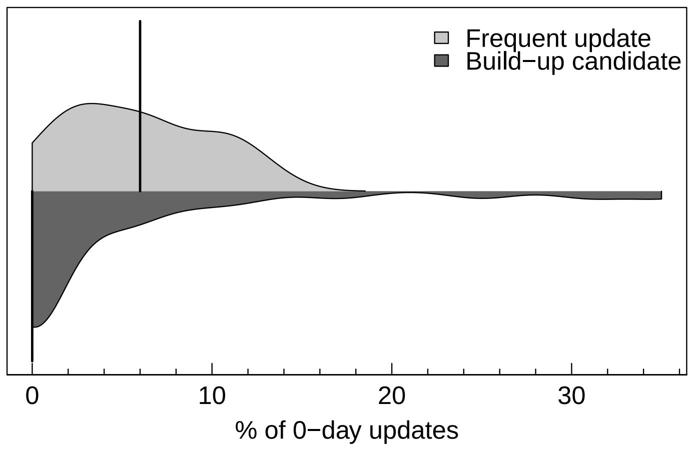

Figure 4.10: Distribution of the percentage of 0-day updates (of all updates) of each studied game. The vertical lines represent the median. The distributions are significantly different with a medium effect size.

4.5.1 Urgent Update Frequency

80% of the studied games have urgent updates. 40 out of 50 (80%) studied games have urgent updates, while the other 10 games all have less than 7 updates (making it difficult to identify urgent updates for these games). The high percentage of games that have urgent updates shows that urgent updates are a common phenomenon across popular games.

Games that use a frequent update strategy tend to have a higher proportion of 0-day updates than games that use a build-up candidate update strategy. As mentioned in Section 4.4, the number of off-cycle updates or hotfixes is not impacted by the choice of update strategy. However, the Wilcoxon rank sum test shows that the difference between the distributions of the percentage of 0-day releases of games using different update strategies is significant, with a medium effect size. Figure 4.10 shows the distribution of the percentage of 0-day updates. 60% of the games that use a build-up candidate update strategy have no 0-day updates, while 93% of the games that use a frequent update strategy have at least one 0-day update. 57% of the games that use a frequent update strategy have at least 5% 0-day updates.

It is interesting to observe that games that follow a build-up candidate update strategy either have very robust updates, i.e., updates that do not require urgent updates, or hold off their fixes until the next update candidate. Another possibility is that the development processes of games that use a build-up candidate update strategy are not suitable for releasing an update so shortly after the previous update (e.g., because the update process is too tedious).

46% of the studied games have self-admitted hotfixes. Table 4.4 shows that 23 out of 50 (46%) studied games have self-admitted hotfixes. In addition, in 12 of these 23 games more than 10% of the updates are self-admitted hotfixes.

Table 4.4 shows that the *DayZ* game and the *Don't Starve Together* game are the games with the highest percentage of self-admitted hotfixes (i.e., more than 40% of the total number of updates). The high percentage of self-admitted hotfixes for the *DayZ* game and the *Don't Starve Together* game can be explained by the fact that these games are early access games[3]. Early access games allow customers to purchase the game during its public beta period while developers continue working on the game. Developers of early access games can receive crucial feedback and bug reports directly from their target community in the earlier state of development. Hence, developers may frequently release self-admitted hotfixes to respond to the received customer feedback. The other early access games that we studied follow a strategy that appears to be less focused on hotfixes, as the percentage of self-admitted hotfixes for those games varies from 0% to 23%. We looked into the early access games and find that the *Rust* game publishes its update notes on Twitter. We manually inspected the Twitter account of

[3]http://store.steampowered.com/earlyaccessfaq/

Rust and observed that its developer releases small updates frequently, often within a week of the previous update, which may be the reason for publishing release notes through informal Twitter updates instead of formal Steam updates.

Although almost half of the studied games have self-admitted hotfixes, Table 4.4 shows that the seven most popular games do not release self-admitted hotfixes. In total, 27 out of 50 games never release self-admitted hotfixes. Moreover, only 10% of the 0-day updates are self-admitted hotfixes. We manually looked into the update notes of 0-day updates which are not self-admitted hotfixes. In most cases, Developers do not give an explanation as to why they are releasing the urgent update within the same day as the previous update. The lack of an explanation and the self-admittance that an update is an urgent update, suggests that developers rarely divulge their botches.

An interesting observation is that while non-game software developers tend to avoid frequent updates because of customer complaints (e.g., Microsoft's "Patch Tuesday" (Microsoft, 2003)), game developers do not seem to care as much about avoiding frequent updates. The explanation could be that the impact of frequent updates on the player of a game is much smaller than on users of non-game software applications, as these are often used in enterprise situations in which updating software requires much effort (e.g., for testing interactions with other applications and the need for carefully planned rollouts of updates). In addition, the Steam platform enforces and manages the update process of games, reducing the required effort of updates of games.

Table 4.7: Reasons given in the update notes for urgent updates (separated by urgent update type, ordered by % of update notes)

0-day updates		**Faster off-cycle updates**		**Self.adm hotfixes**		**All urgent updates**	
Reason	**%**	**Reason**	**%**	**Reason**	**%**	**Reason**	**%**
Functionality	59	Functionality	71	Functionality	64	Functionality	64
CrashingGame	32	UserInteraction	49	CrashingGame	46	CrashingGame	39
Visual	26	Visual	37	Visual	35	Visual	32
UserInteraction	22	RuleChange	34	RuleLoophole	27	UserInteraction	27
RuleChange	21	CrashingGame	29	UserInteraction	25	RuleChange	25
Content	19	RuleLoophole	24	RuleChange	25	RuleLoophole	23
RuleLoophole	18	Performance	24	Performance	25	Performance	22
Performance	16	Content	20	Content	20	Content	19
Sound	9	Sound	15	Sound	11	Sound	11
Localization	4	Localization	5	Localization	4	Localization	4
Security	3	Security	0	Security	4	Security	3

* Note that these percentages do not add up to 100% as multiple reasons can be given in the update notes of a single update.

4.5.2 Reasons for Releasing Urgent Updates

36% of the urgent updates are released to make changes to the rules of a game. Table 4.7 shows the frequency of each reason given in the update notes of urgent updates. While the identified most commonly-given reasons for releasing urgent updates apply to software in general, the rule-changing urgent updates are specific to games. We calculated that 36% of the urgent updates are labelled as RuleLoophole or RuleChange (or both). On the one hand, loopholes in the rules (23%) must be rapidly fixed in order to prevent cheating. For example, in the *Brawlhalla* game, an urgent update was released to address the following: *"Dodging in the same direction of an item will not provide dodge forgiveness immunity. Ex: Dodging away from a throw means you will be immediately vulnerable to a weapon thrown directly at you."* On the other hand, developers can decide to make the game more playable by slightly changing the rules of a game by modifying the value of particular parameter settings (25%). For example, in

Table 4.8: Reasons given in the update notes for urgent updates (seperated by update strategy, ordered by % of update notes)

Frequent update		**Build-up candidate**	
Reason	**%**	**Reason**	**%**
Functionality	61	Functionality	72
CrashingGame	39	RuleChange	38
Visual	31	CrashingGame	35
UserInteraction	26	Visual	35
RuleChange	21	RuleLoophole	35
RuleLoophole	20	UserInteraction	29
Performance	20	Performance	26
Content	18	Content	23
Sound	10	Sound	14
Localization	3	Localization	5
Security	2	Security	5

* Note that these percentages do not add up to 100% as multiple reasons can be given in the update notes of a single update.

the same game, an urgent update was released to make items spawn faster after a game starts: *"Community Request: - Lowered the delay at the start of the game until items begin spawning by 750ms"*. Both of the aforementioned urgent updates for *Brawlhalla* were released in response to player requests.

Feature malfunctions, crashing games and visual bugs are the most commonly given reasons for releasing urgent updates. Table 4.7 shows that 64% of the update notes mention a functional issue as a reason for releasing the urgent update. Moreover, a functional issue is also the top reason for releasing the three kinds of urgent updates. While the other reasons that we identified relate to issues that negatively impact the gaming experience, feature malfunctions, crashing games and visual bugs are issues that can actually render a game unplayable.

The major difference between the reasons that are given across the two update strategies is that games that use a build-up candidate update strategy release a higher percentage of urgent updates because of a RuleChange. Table 4.8 compares the frequency of each reason across the two update strategies. As stated by the *League of Legends* game, an imbalance in the rules of a game is a type of issue that requires an immediate fix (Lesensmer, 2013), as it directly affects gameplay. Because the *days-between-updates* is higher for games that use a build-up candidate update strategy, these games need to release an urgent update to immediately address a RuleChange issue, while games that use a frequent update strategy are more likely to be able to include the fix in a regular update.

Localization and security are the least commonly-given reasons for releasing urgent updates. Although it is understandable that localization issues are not deemed urgent, in only 3% of the analyzed update notes, security is given as a reason for releasing the urgent update. This may seem as a surprisingly low number, considering the possible impact of security vulnerabilities and the urgent need for a quick solution. In online games, security vulnerabilities are often related to cheating. Cheating allows players to break game rules, which in turn may lead to financial benefit (McGraw and Hoglund, 2007), e.g. by illegally obtaining access to high-level gaming profiles or rare in-game items. Motoyama et al. (2011) show that Steam accounts are the second most popular trading item on underground forums, beating credit cards in popularity. In addition, hacking Steam accounts has been offered as an on-demand service on underground forums (Stone, 2016). We expect that the low number of security-related urgent updates is because developers do not give security as a reason, but explain such urgent updates instead as for example, fixes for functional issues or loopholes in the rules of the game. Another possible explanation is that some urgent updates that are related to security issues can be fixed (or at least temporarily addressed) through server-side changes only. Hence, there are no update notes for these urgent updates as there is no downloadable component (Wow Wiki, 2009).

Not all urgent updates address issues that are caused by the previous update. 12 (4%) of the studied update notes advertise the release of new downloadable content. A possible explanation is that the development of new downloadable content is done in parallel with the regular update cycle of games.

In addition, the developers of the *Rust* game explain that an unexpected urgent update is due to a request from the Steam platform to add a censorship module to the game, as Steam does not want players to *"flood the rest of Steam with pictures of cavemen genitalia"* (RUBAT, 2013). The *Rust* case suggests that external pressure to the developers can also be a reason of interrupting their usual update cycle.

4.5.3 Comparison with Prior Work

As mentioned in Chapter 3, Lewis et al. identified 11 types of failures in video games by surveying game failure videos on YouTube. Table 4.9 shows a mapping of Lewis et al.'s taxonomy and the reasons that we observed for releasing urgent updates. An interesting observation is that some of the reasons that we observed for releasing urgent updates are difficult to observe from game failure videos (e.g., CrashingGame and Security). Therefore, Lewis et al.'s and our taxonomy are complementary to each other.

Summary: 80% of the studied games have urgent updates. Games that follow a frequent update strategy tend to have a higher proportion of 0-day updates. Feature malfunctions, crashing games and visual bugs are the most commonly-given reasons for releasing urgent updates.

Table 4.9: Mapping between Lewis et al.'s categories (Lewis et al., 2010) and the reasons for releasing urgent updates that are identified in this chapter

This chapter	**Lewis et al. (2010)**
Functional	Invalid value change, Artificial stupidity, Information, Action, Invalid position over time, Invalid context state over time, Interrupted event
CrashingGame	-[1]
RuleLoophole	Invalid value change, Object out of bounds, Action
RuleChange	Invalid event occurrence over time
Content	Object out of bounds
Visual	Invalid graphical representation, Information, Implementation response issues
Sound	Interrupted event
UserInteraction	-[1]
Performance	Implementation response issues
Localization	-[1]
Security	-[1]

1. We do not find any Lewis et al.'s category which maps this reason.

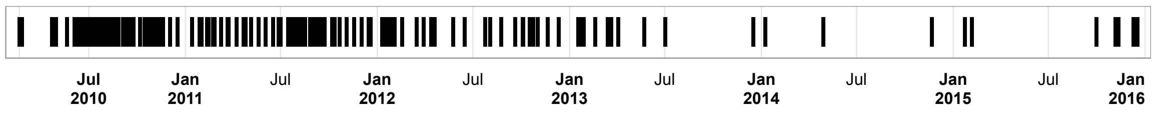

Figure 4.11: Update timeline of the *Left 4 Dead 2* game. Each vertical line represents an update.

4.5.4 Discussion of the Possible Reasons for Slower Off-cycle Updates

As mentioned in Section 4.4, slower off-cycle updates are commonly identified in the studied games. In this section, we discuss the possible reasons for slower off-cycle updates. We studied the update timeline of all studied games and we observed that many games take longer to release an update as the age of the game increases. Figure 4.11 shows the update timeline of the *Left 4 Dead 2* game as an example. As shown in the figure, the game updates very frequently at the beginning of its lifetime. However, after

July 2013 (approximately three years after the initial release), the *days-between-updates* significantly increases. Hence, most updates after July 2013 are slower off-cycle updates. A possible explanation is that, after a certain time period, games reach maturity and require maintenance updates only. Another possible explanation is that a game developer focuses on releasing updates for other games (e.g., a new version of the game) and releases only updates that are necessary to keep the game playable.

4.6 Threats to Validity

In this section, we present the threats to the validity of our findings.

4.6.1 Internal Validity

A threat to the validity of our findings is that it is not necessary for game developers to publish update notes for a game update to one of the Steam channels. Hence, all numbers that we give in this chapter may be low bound estimates of the actual number of updates.

In our study, we assumed that a problem which later leads to an urgent update is introduced by the update preceding that urgent update. While this assumption may threaten the validity of our findings, we encountered only a very small portion (i.e., approximately 4%, the downloadable content and the censorship updates) of urgent updates that exhibited proof against this assumption during our analysis.

4.6.2 External Validity

In our empirical study, we studied the 50 most popular games on Steam. The findings of our study may not generalize to other games with different distribution mechanisms. However, as stated in Chapter 2, Steam is the largest digital distribution platform for PC gaming. Hence, popular Steam games are representative for a large number of games.

4.6.3 Construct Validity

We identified off-cycle updates with a threshold of 2 times MAD[4]. Although we conducted a preliminary experiment to find the threshold that works best for our data, it is possible that some off-cycle updates were not identified by this threshold.

We manually validated our approach for collecting update notes for self-admitted hotfixes and observed that our approach has a precision of 88% and a recall of 87%, as described in Section 4.3.2.

4.7 Chapter Summary

In this chapter, we studied the urgent updates of popular games on Steam. Urgent updates fix issues that are deemed critical enough to not be left unfixed until the next regular update.

We conducted an empirical study on 2,419 update notes of the 50 most popular games on the Steam platform. We used update notes to 1) identify the update strategy

[4]Median Absolute Deviation, see Section 4.3.

that is followed by each game, 2) identify and study urgent updates and 3) study the reasons for releasing urgent updates. The most important findings of our study are:

1. 80% of the studied games have urgent updates. Games that use a frequent update strategy have a higher proportion of 0-day updates than games that follow a build-up candidate update strategy.
2. 46% of the studied games have self-admitted hotfixes. Only 10% of the 0-day updates are self-admitted hotfixes, which suggests that developers rarely divulge their mistakes.
3. 36% of the urgent updates are released to make changes to the rules of a game.
4. Feature malfunctions, crashing games and visual bugs are the most commonly given reasons for releasing urgent updates.

The most important contribution of this chapter is the finding that the choice of update strategy seems to affect the proportion of 0-day updates that developers have to release. We observed that games that release frequently also release a higher proportion of 0-day updates than games that use a traditional build-up candidate update strategy. Our findings are consistent with the findings of Souza et al. (2015), who observed that releasing frequently leads to a higher proportion of patches that must be reverted.

Prior work (Khomh et al., 2012, 2015; da Costa et al., 2016) on update strategies focused mostly on the Mozilla Firefox project, in which the update strategy changed from traditional build-up candidate updates to frequent updates (i.e., every six weeks). In this chapter, we showed that most games update much more frequently than once every six weeks, a phenomenon that was recently observed for mobile apps (McIlroy et al., 2016a). The unique distribution mechanism (e.g. online store) of games and mobile apps allows developers to release updates for their software at an increasingly rapid pace. Future research efforts need to carefully reconsider how such rapid pace of updating software influences our well-established understandings of software engineering practices and theories.

In this chapter, we observed that early access games have a high percentage of self-admitted hotfixes (see Section 4.5.1). In the following chapter we study early access games on the Steam platform to understand the characteristics, advantages and limitations of the early access model.

CHAPTER 5

Studying Early Access Games on the Steam Platform

In Chapter 4, we studied the urgent updates of popular games on the Steam platform, and observed that early access games have a high percentage of self-admitted hotfixes. "Early access" is a release strategy for software that allows consumers to purchase an unfinished version of the software. In turn, consumers can influence the software development process by giving developers early feedback. This early access model is considered a success by the game development community as several games using this approach have gained a large user base (i.e., owners) and high sales. On the other hand, the benefits of the early access model have been questioned as well. In this chapter, we conduct an empirical study on 1,182 Early Access Games (EAGs) on the Steam platform to understand the characteristics, advantages and limitations of the early access model. We observe that on the one hand, developers update their games more frequently in the early access stage. On the other hand, the percentage of players that review a game during its early access stage is lower than the percentage of players that review the game after it leaves the early access stage. However, the average rating of the reviews is much higher during the early access stage, suggesting that players are more tolerant of imperfections in the early access stage. Based on our findings, we suggest game developers to use the early access model as a method for eliciting early feedback and more positive reviews to attract additional new players.

An earlier version of this chapter is published in the Empirical Software Engineering Journal (EMSE) (Lin et al., 2018a).

5.1 Introduction

EVERY year, 70% of the software development projects do not deliver the expected product (Johnson, 2000), despite the expenditure of $275 billion on software development projects in the U.S. alone (Johnson, 1999). The failures include total failures, time and budget overruns, and unmet user requirements (Johnson, 2000).

In order to prevent the problems of overrunning budget and time, and to better meet the user requirements, a *public beta-release* release strategy is commonly used by software developers. As early as in 1984, a “pioneer edition" of the WordVision word processor for the IBM PC was available for early customers to purchase (Manes, 1984). Microsoft launched “Office Insider” program in late 2015, which allows customers to get early access to the latest Office features and provide feedback (Foley, 2015). Another example is the Minecraft game. Available since 2009, Minecraft stayed in public beta until 2011 (Minecraft, 2016). As the sales of the public beta version increased, its developer was able to quit his day job to work on Minecraft full-time (McDougall, 2010). During the beta stage, Minecraft raised over $33 million from the public beta sales while accumulating over 1.8 million players (Orland, 2011).

Inspired by the successful application of the public beta release strategy in games such as Minecraft, Steam started to offer game developers the opportunity to release their games as public betas in March 2013. These so-called “Early Access Games" (EAGs), allow customers to purchase the public beta version of a game while developers continue working on the game. Developers of EAGs receive crucial feedback and bug reports directly from their target community, while players have the opportunity to be the first to play new games and get involved with games as they evolve. Hence, as declared by Steam, early access is "*the way games should be made*" (Valve, 2016a).

The early access model made a name for itself through several successful games, such as the *DayZ* game[1]. The multiplayer survival-based game reached 400,000 sales during its first week as an EAG, according to its developer Bohemia (EuroGamer, 2014).

However, the benefits of the early access model have been questioned as well. For instance, the *Spacebase DF-9* game[2] abandoned the early access stage unexpectedly as the funds raised during that early access stage were not sufficient to continue the development process. As a result, many promised features were left unimplemented, disappointing many players of the game. The game received 77% negative reviews (Valve, 2018). Shortly after abandoning the early access stage and terminating the development, twelve employees were laid off including the programmer and project lead (LeBreton, 2014). The developer stated that all funds raised during the early access stage went into the development of the game, but that eventually the studio was spending more than it was making (PCGamer, 2014).

Along with the aforementioned failure of the early access model, the debate of whether early access is as good as expected has been raised. One year after the release of the Steam Early Access Release Platform, Walker (2014) calculated that only 25% of the EAGs have left the early access stage. Recently, Allen (2016) manually investigated the first 50 released EAGs and warned people that the early access model may be "a ticking time bomb", as the development of 32 (64%) of the first 50 EAGs was either abandoned or inactive. Allen states that the early access model currently has a bad reputation and is leading games to a "Development Hell", and calls for a systematic

[1]http://store.steampowered.com/app/221100/
[2]http://store.steampowered.com/app/246090/

in-depth study of all EAGs to examine the opportunities and risks that lie behind the early access model.

In order to get a better understanding of the impact and limitations of the early access model, we conducted such an in-depth empirical study on EAGs on the Steam platform. The study aims at providing developers with an understanding of the characteristics of the early access model, the degree of interaction between developers and players of EAGs with the Steam platform, and the tolerance of players of the quality of EAGs. Additionally, based on these results, we provided suggestions for developers to make the optimal use of this novel release strategy. In particular, we addressed the following three research questions (RQs):

RQ1: What are the characteristics of the early access model?

Currently, 15% of the games on Steam use the early access model. The early access model tends to appeal mostly to individuals or small studios for releasing their indie games. However, using the early access model is not a guarantee for collecting enough funds to continue the development of a game.

RQ2: How do developers and players of EAGs interact with the Steam platform?

Developers update a game more frequently during its early access stage. Players post less reviews, however players have more discussion posts in the early access stage.

RQ3: How tolerant are players of the quality of EAGs?

Players of EAGs tend to be more tolerant of the quality of a game during its early access stage. While players tend to post less reviews within the early access stage, 89% of EAGs receive an equally or more positive review rate in their early access stage. In addition, developers do not need to rush into releasing their games, as the tolerance of players does not correlate with the length of the early access stage.

5.2 Background on the Study of Early Access Games

This section describes the mechanism of Steam's early access release platform, the differences between crowd-funding and early access, and the related work.

5.2.1 Steam Early Access Release Platform (SEARP)

The SEARP was launched on March 20, 2013, with 12 game titles available initially (Welch, 2013). The platform allows developers to release unfinished, yet playable games, so-called *Early Access Games* (EAGs). By purchasing an EAG, players are allowed to download and play that game in its current state and as it evolves, even after the game leaves the early access stage.

The SEARP provides developers with early access sales and distribution mechanisms. The developers of EAGs have the freedom to determine when to move a game out of the early access stage. In addition, developers have the freedom to increase or decrease the price of their game at any time. Players are aware of the risk that a game may be incomplete, buggy, or unfinished when purchasing an EAG. All reviews posted during the early access stage of a game are tagged as "early access review", hence they can be distinguished from the reviews that are posted after leaving the early access stage.

5.2.2 Crowdfunding vs. Early Access

Crowdfunding is the practice of funding a project or venture by raising small amounts of money from a large number of supporters, typically via the Internet (Oxford, 2016). Many games use a type of crowd-funding model called “Reward Crowdfunding” to support the game development costs, by which the developers pre-sell the product to launch the project without incurring debt (Clifford, 2014).

There exist similarities between the crowdfunding and early access model, as both models raise funds by selling products before their completion. However, the differences between early access and crowdfunding are worth noting. Although many crowd-funded games promise to offer access to alpha or beta versions of the game, no playable version usually exists during the initial crowd-funding campaign. All Steam EAGs offer an immediately playable version of the unfinished game to customers. However, in both models paying customers take the risk that they may never see a final release of the game.

It is worth noting that in order to minimize the risk, Valve (the company to which Steam belongs) tightened the SEARP rules for developers on November, 2014, stating that SEARP is “meant to be a place for games that are in a playable alpha or beta state, are worth the current value of the playable build, and the developer plans to continue to develop for release” (Yin-Poole, 2014). The newly added rules include “Don’t launch in Early Access if you can’t afford to develop with very few or no sales” and “Make sure you set expectations properly everywhere you talk about your game”, which seem to directly target the failure of the aforementioned *Spacebase DF-9* game, a month before releasing the new rules. We further discuss the learnt lessons from the *Spacebase DF-9* game failure in Section 5.5.

5.2.3 Related Work

In this section, we discuss prior research related to our study. Most of the work that is related to our study focused on early releases in software or on user involvement in software development.

Beta Releases in Software

Several studies regarding the perpetual beta (i.e., where the product is developed in the open, with new features added on a monthly, weekly, or even daily basis) in software have been done. O'reilly (2007) pointed out that one of the fundamental changes in the software release cycle in Web 2.0 is the use of the perpetual beta in which users are treated as co-developers. Ullrich et al. (2008) stated that the perpetual beta increases the value a user gets from using the service. Developers using the perpetual beta release model are interested in feedback and are open to suggestions.

Al-Ani et al. (2008) observed that traditional software development models either impose too tight (i.e., costly and infeasible) or too loose (i.e., not efficient) restrictions on user participation in the development process. They suggested a continuous form of participation is the most efficient form of participation.Maalej et al. (2009) proposed a continuous and context-aware approach for communicating user input to engineering teams.

Our study is one of the first to study beta releases (i.e. the early access model) in games.

Interaction between Users and Developers

Several studies on interaction between users and developers exist in literature. One of the topics is about the participatory design in games. Jacobs and Sihvonen (2011) distinguished two forms of participatory design between players and developers that are commonly implemented: direct participatory design (connecting with a small number of highly active players) and silent participatory design (silently log all actions from all players). Jacobs and Sihvonen used the example of Facebook games that are developed by the Zynga company to show that these two forms can be implemented in a perpetual beta. However, Jacobs and Sihvonen warned that once the game development is centered around player feedback, in the end, the game environment will become unbalanced as players only design the game from a player perspective (wanting what is scarce in the game).

Löwgren and Stolterman (2004) claimed that participatory design is a mutual learning process between users and designers and it is not only users participating in design, but also designers participating in use. Taylor (2006) explored relationships between players and developers of massively multiplayer online games (MMOG). Löwgren and Stolterman stated that *"at the heart of games is a complex negotiation between what the player might like to do and what they must or should do."*

Other examples include user involvement in software development. Kujala (2003) conducted a study of the benefits and challenges of user involvement. The study claimed that user involvement generally has a positive effect, especially on user satisfaction. However, the role of users must be carefully considered, as developers and users tend to have difficulties in communicating, and user groups may have conflicts.

Table 5.1: Dataset description of our study of early access games

# of games	8,025
# of EAGs	1,182
# of current EAGs	786
# of former EAGs	396
# of news updates	104,236
# of release notes	38,249
# of EAG news updates	31,916
# of EAG release notes	16,780
# of reviews	12,338,364
# of early access reviews	1,564,574
# of discussion posts of former EAGs	801,128

Damodaran (1996) provided guidelines for user involvement in the system design process. Gallivan and Keil (2003) proposed a process model that delineates the four stages of communication between users and software developers, and advised researchers and practitioners on how to leverage the potential benefits of user participation, rather than take the benefits for granted.

The early access model has a potential to improve user involvement in game development. This chapter makes an initial step by exploring how users and game developers interact with the Steam platform.

5.3 Our Methodology for Studying Early Access Games

This section introduces the methodology of our empirical study of EAGs. We detail how we extract and process data. Table 5.1 presents the description of our collected dataset. Figure 5.1 gives an overview of our methodology.

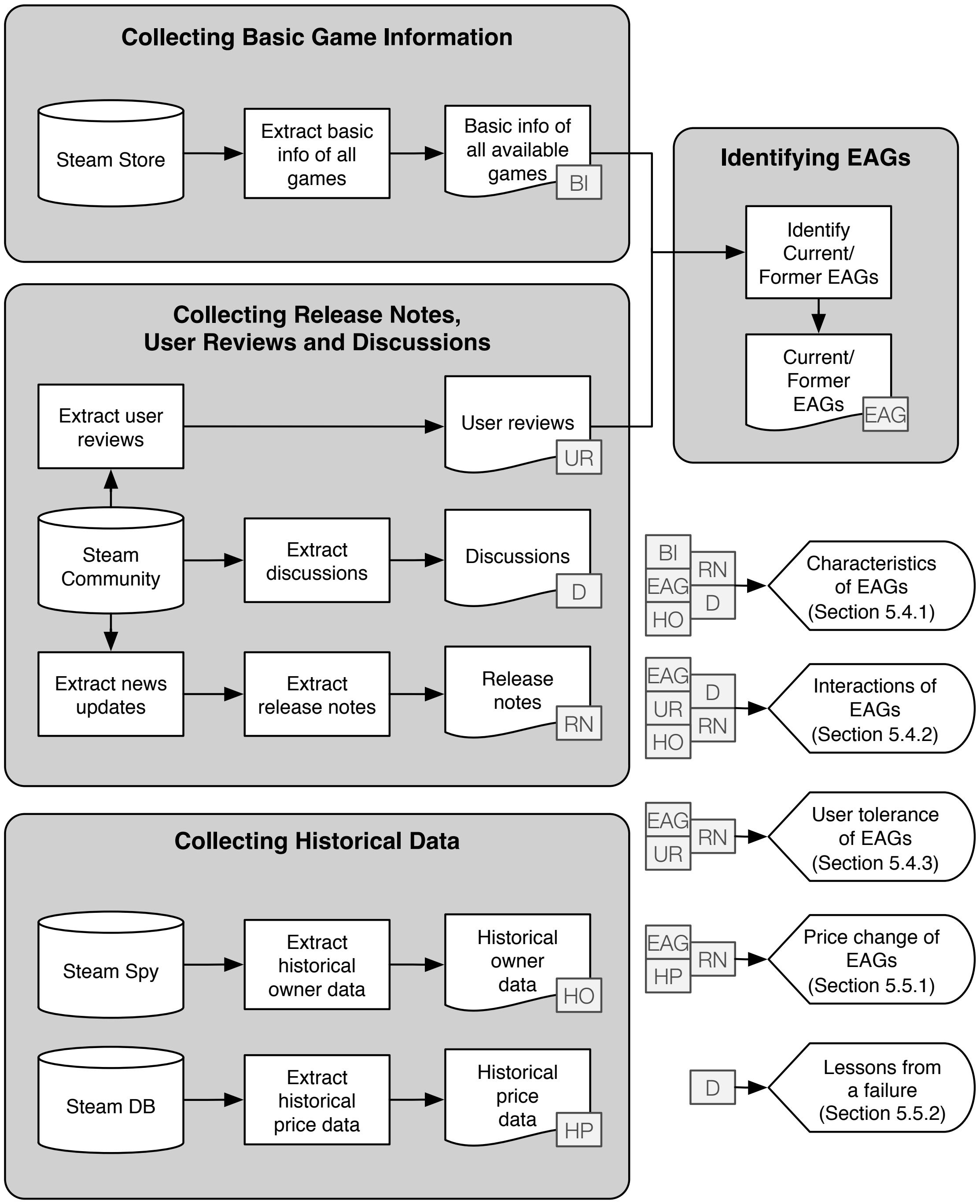
Collecting Basic Game Information
Steam Store
Extract basic info of all games
Basic info of all available games
BI
Identifying EAGs
Identify Current/ Former EAGs
Current/ Former EAGs
EAG
Collecting Release Notes, User Reviews and Discussions
Extract user reviews
User reviews
UR
Steam Community
Extract discussions
Discussions
D
Extract news updates
Extract release notes
Release notes
RN
BI EAG HO RN D
Characteristics of EAGs (Section 5.4.1)
EAG UR HO D RN
Interactions of EAGs (Section 5.4.2)
EAG UR RN
User tolerance of EAGs (Section 5.4.3)
Collecting Historical Data
Steam Spy
Extract historical owner data
Historical owner data
HO
EAG HP RN
Price change of EAGs (Section 5.5.1)
Steam DB
Extract historical price data
Historical price data
HP
D
Lessons from a failure (Section 5.5.2)

Figure 5.1: Overview of our study of early access games

5.3.1 Collecting Basic Game Information

We developed a customized crawler to take a snapshot of all the 8,025 games that are available in the Steam Store on March 7th, 2016. We collected the title, developer, publisher, tags, genres, and current early access status (i.e. whether the game is in the early access stage or not) of each game. The tags of each game are specified by its players, while the genres of a game are specified by its developer.

5.3.2 Collecting Release Notes, User Reviews and Discussions

In order to study the update frequency of games, we used the accompanying release notes that are posted on channels in the Steam Community. We used the process described in Section 4.3 to extract the release notes of each game from the channels. We extracted all 104,236 news updates for all available games on March 7th, 2016 using a custom-written crawler, and identified 38,249 release notes for all 8,025 games.

We extracted all the reviews for each game from the Steam Community. There are in total 12,338,364 reviews across all supported natural languages. We also extracted all the threads from the discussion forums on the Steam Community for all EAGs that have left the early access stage. We extracted discussion posts for all the EAGs that left the early access stage only, because doing so allows us to study the difference in interaction between players and the Steam platform through the discussion forums within and after leaving the early access stage. We extracted the discussion posts (i.e., a message by a user or developer) from all the subforums except for the *Trading* subforum, because the discussion posts in *Trading* do not contain player feedback, instead they discuss trades among players. In total, we extracted 801,128 discussion posts.

5.3.3 Identifying EAGs

Because Steam does not provide a list of EAGs, we used the following approach to identify them.

Current EAGs

If a game is currently in the early access stage, its Steam Store page would explicitly state that this game is an EAG. We used the existence of this statement to identify games that are currently in the early access stage. These games are in the remainder of this study referred to as current EAGs.

Former EAGs

Because the Steam Store does not explicitly identify games that already left the early access stage, we used the existence of early access reviews (i.e. reviews with the "early access review" tag) to get a minimal indication of whether a game used the early access model at some point. Such identified games are in the remainder of this study referred to as former EAGs.

5.3.4 Collecting Historical Data

We extracted the history of the number of owners since March 20th, 2015 for all games from Steam Spy (Galyonkin, 2018), a third-party project which continuously monitors the Steam platform. People own a game when they buy the game on Steam, in retail and then activate on Steam, or when they receive the game through a promotion or as a gift (Galyonkin, 2018). Different from owners, the players of a game are people who play the game during a specific time range. Hence, the number of owners is not necessarily the same as the number of players in a day. However, as we only use the number of owners in this study, we use players and owners interchangeably in the remainder of this chapter.

Due to the large quantity of data that is collected from the Steam platform, the crawl cannot be done instantly. In fact, the crawling process started on March 7th and ended on March 19th, after which the number of owners' data was crawled from Steam Spy on March 20th. We used the data from both sources up to March 7th to ensure that we study the same time frame for all games.

As often happens in the gaming industry, all the data needed to track sales figures on Steam are not publicly available. Nevertheless, Steam Spy estimates the number of owners of a game (Orland, 2014). The method uses information from user profile pages on the Steam Community, which shows the games that a user owns. Theoretically, by crawling the profile pages for all users, we can calculate the accurate ownership statistics. Practically, with about 172 million users (and growing every day) on Steam, it is hard to have the computing power needed to churn through all profile pages in a timely manner. Steam Spy randomly crawls a representative sample of user profile pages to estimate the number of owners. To be more accurate, Steam Spy uses a three-day rolling sample to generate the final reported numbers of owners, i.e., every day, the data from three days prior are replaced by newly-crawled data. About 1,700,000 randomly-selected profiles are crawled every three days.

We also extracted the price history since November 27th, 2014 for all games from the Steam DB project (Djundik and Benjamins, 2016), another third-party project that monitors the Steam platform. We used the price of a game in U.S. Dollar in our study.

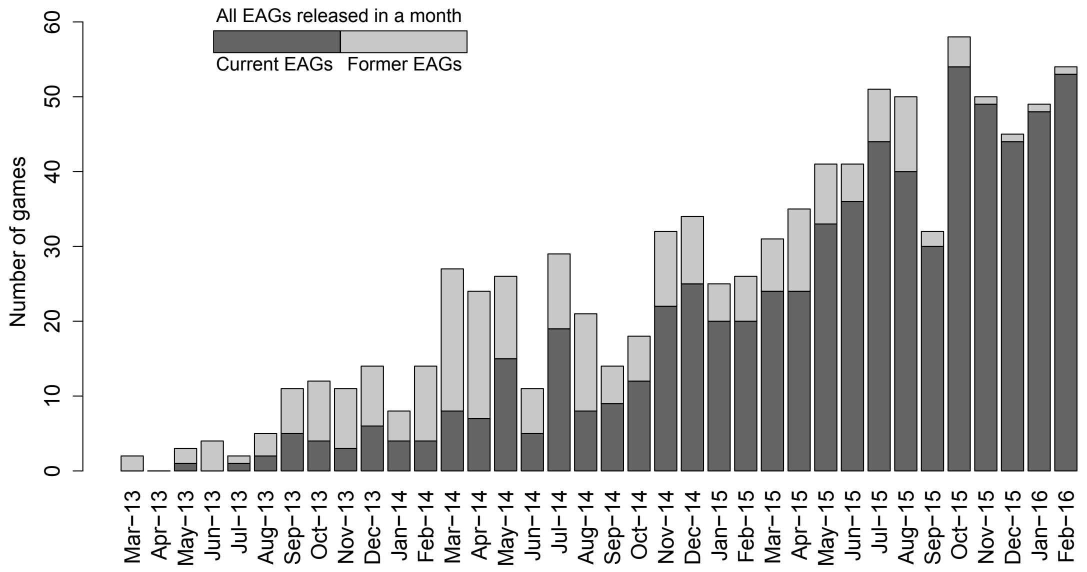

Figure 5.2: The number of EAGs that are released since the start of the SEARP. The darker part represents the number of EAGs that are released in that month that are still in the early access stage at the time of our data collection. The lighter part represents the number of EAGs that are released in that month that have left the early access stage at the time of our data collection.

5.4 Results of our Study of Early Access Games (EAGs)

This section presents the results of our empirical study on EAGs.

5.4.1 RQ1: What are the characteristics of the early access model?

Motivation: We studied the characteristics of the early access model. As few prior studies have focused on the early access model in the gaming industry (Walker, 2014; Allen, 2016), it is essential to have a general understanding of the current status of the model. The results that are described in this section motivate the remainder of this chapter.

Approach: We analyzed the popularity of the early access model by studying the number of games on the SEARP and the number of owners of EAGs. We plotted the proportions of games that are released as EAGs by each developer. In addition, we calculated the length of the early access stage, and studied the drivers for short and long early access stages.

In order to get the length of the early access stage of each game, we manually checked the release notes for each game and identified the release notes that describe the availability of the game on Steam and the game leaving the early access stage. We used the number of days between the publication dates of these two release notes as the length of the early access stage of a game. As mentioned in Chapter 2, it is not mandatory for developers to publish these release notes. We were able to identify 227 (out of 396) games which have both release notes for entering and leaving the early access stage.

Findings: **15% of the games on Steam make use of the early access model and its popularity is growing.** Of the 8,025 games that are available on Steam, 786 games are current EAGs, and 396 games are former EAGs. As a result, 1,182 (15%) games are or were making use of the early access model.

Figure 5.2 shows the popularity of the early access model. The figure clearly shows that there is a growing trend of popularity in the use of the early access model. With 64 games released on the Steam early access platform in 2013, and 485 games newly available through early access in 2015, the model shows a 660% increase in the absolute number of releases.

The increasing trend in popularity is confirmed by Figure 5.3, which shows the ratio of the number of EAGs that are released each month and the total number of games

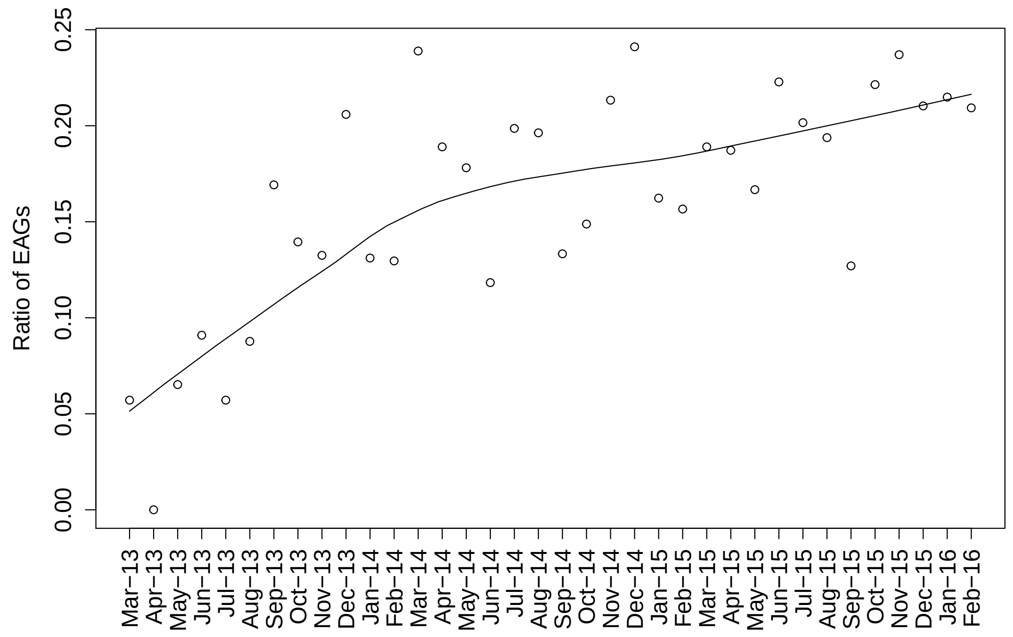

Figure 5.3: The ratio of the number of released EAGs each month and the total number of games that are released in that month. The smooth curve is computed using a Local Polynomial Regression Fitting (Chambers and Hastie, 1991).

that are released in that month. The ratio increases from approximately 0.05 to 0.20 in early 2016.

25% of the EAGs have more than 48 thousand owners, with almost 29 million owners for one of the studied EAGs. Figure 5.4 shows the distribution of the number of owners of EAGs. A considerable number (62%) of the EAGs has been available for less than a year, leading to a median number of owners of 11,270. Moreover, 25% of the EAGs have more than 47,950 owners, with 43 (3%) of the EAGs having more than 1 million owners. The EAG with the highest number of owners, the *Killing Floor 2* game[3] has 28,878,959 owners.

[3] http://store.steampowered.com/app/232090/

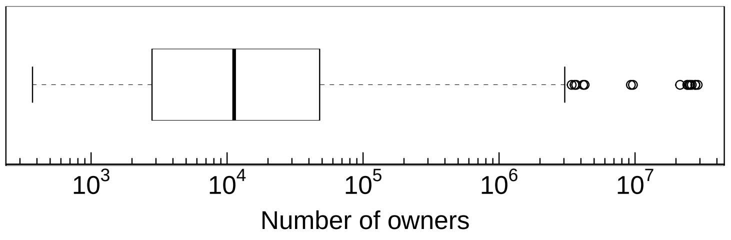

Figure 5.4: Distribution of the number of owners of EAGs.

34% of all EAGs have left the early access stage. This number can partly be explained by the recency of an early access release. However, EAGs from 2013 do not have a considerably higher percentage of leaving the early access stage. Only 162 (50%) of the 322 EAGs that were available before 2014 have left the early access stage. Hence, customers are taking the risk that an EAG will possibly spend a long time in development (or that the game will even fail to leave the early access stage eventually).

Walker (2014) has conducted a similar calculation in 2014, and obtained a percentage of 25% instead of 34%. We contacted Walker and he kindly provided a list of games that were studied in his article on early access games. After comparing our dataset with his list, we observed that only 266 of the 366 games in his list were available on Steam at the time at which we collected our data. Hence, one possible explanation of the 9% growth is that in the past two years some EAGs were removed from the Steam store. Therefore, these games were no longer available at the time that we collected our dataset.

88% of the EAGs are *indie* games, indicating that most EAGs are developed by individual developers or small studios. Table 5.2 shows the top 10 developer-defined genres for EAGs. For both EAGs and non-EAGs, indie games are the largest genre. However, only half of the non-EAGs are defined as indie games, while 88% of the EAGs are indie games.

Table 5.2: Top 10 genres for EAGs and non-EAGs

EAGs			**Non-EAGs**		
Genre	**# of games**	**%***	**Genre**	**# of games**	**%***
Indie	1,046	88.49	Indie	3,863	56.45
Early Access	783	66.24	Action	3,246	47.44
Action	752	63.62	Adventure	2,859	41.78
Adventure	499	42.22	Singleplayer	2,123	31.02
Strategy	403	34.09	Casual	2,003	29.27
RPG	363	30.71	Strategy	1,663	24.30
Simulation	348	29.44	RPG	1,318	19.26
Multiplayer	322	27.24	Simulation	1,173	17.14
Singleplayer	296	25.04	Multiplayer	1,084	15.84
Casual	231	19.54	Puzzle	906	13.24

* Note that these percentages do not add up to 100% as developers can assign multiple genres to their games.

To the best of our knowledge, there is no official definition of what an "indie" game involves. We used the universal definition as concluded by Stern (2012): *"A game that is both (a) developed to completion without any publisher or licensor interference, and (b) created by a single developer or a small team."* We assumed that games classified under the "indie" genre on Steam follow this definition.

To validate this definition, we extracted 4,927 unique developers from the basic information of all 8,025 games and counted the number of games that are released by each developer. Figure 5.5 shows the relation between the number of games and the percentage of EAGs that are developed by the same developer. When calculating the percentage of EAGs, we manually filtered out the games that are released before March 20, 2013 (i.e., the start date of the SEARP), the games that are re-released back to Steam, and the games that are content packs for existing games, as logically these games did not have the chance to be released as EAGs. The developers that do not have any game released after March 20, 2013 are also not shown in the figure. Figure 5.5 indicates that,

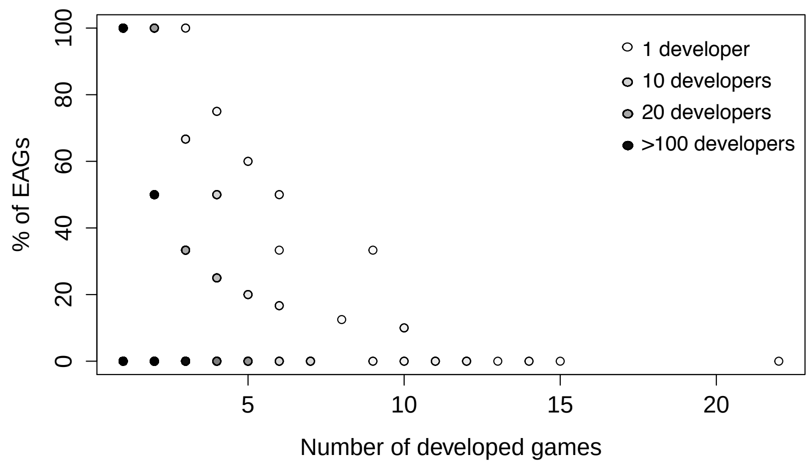

Figure 5.5: Relation between the number of games and % of EAGs that are developed by the same developer (darker dots represent a larger number of developers with that relation)

Table 5.3: The number of EAGs per development studio

Number of EAGs	Number of Studios
0	3,814
1	1,062
2	41
3	9
4	1
Total	4,927

as developers release more games, the percentage of EAGs decreases, indicating that EAGs are mainly developed by individuals or small studios with less games.

Table 5.3 shows the number of EAGs per development studio. Table 5.3 shows that most studios have zero or one EAGs.

Most former EAGs have spent less than a year in the early access stage. Figure 5.6 shows the distribution of the length of the early access stage for former EAGs. 160 of 227 (70%) former EAG spend less than 365 days, i.e. a year, in the early access stage, with a median of 225 days. The longest early access stage record, 929 days, is kept by

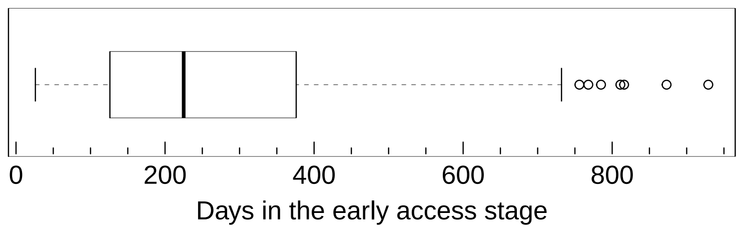

Figure 5.6: Distribution of the days in the early access stage for former EAGs.

the *Prison Architect* game[4]. We manually checked the discussions between developers and players on discussion forums in the Steam Community of games that are more than 800 days in the early access stage, and identified the following reasons as claimed by developers for the long length of the early access stage:

1. A lack of developers in the team, or a lack of funds to hire developers for the team (*Grim Dawn* (Medierra, 2015)).

2. A lack of experience or specific skills by the developers (e.g., UI art) (*Underrail* (Logfeller, 2014), *Grim Dawn* (Medierra, 2015)).

3. Difficulties of estimating the full budget, which causes the delay of hiring more developers (*Grim Dawn* (Medierra, 2015)).

4. A refusal to launch the game until it reaches a very high standard with which the developers themselves are satisfied (*Edge of Space* (LadyAijou, 2015, 2014)).

In addition, we manually checked the release notes of the ten EAGs with the shortest length in the early access stage. The developers of two of such games gave the reason for the short length of the early access stage, while the other eight EAGs did not give a reason. The developers of the *Parcel* game[5], which only stayed in the early access

[4]http://store.steampowered.com/app/233450/
[5]http://store.steampowered.com/app/316080/

stage for 26 days, stated that they had gone over budget and that the early access model failed as a funding channel for them (Takkunen, 2015). The developers of the *RONIN* game[6], which spent 34 days in the early access stage, explained that the game had already been tested by beta testers, other developers and third-party testing studios, and their goal was to perfect the game by using the honest feedback that is gathered in the early access stage (Waclawek, 2015).

Summary: Currently, 15% of the games on Steam use the early access model. The early access model tends to appeal mostly to individuals or small studios to release their indie games. However, using the early access model is not a guarantee for collecting enough funds to continue the development of a game.

5.4.2 RQ2: How do developers and players of EAGs interact with the Steam platform?

Motivation: One of the major benefits of the early access model for developers is that it is possible to get early feedback on a game, for example, through reviews that players post on the Steam platform. As early access players should be deeply involved in the development process as claimed by Steam (Valve, 2016a), we expected to see a stronger interaction of players with the Steam platform in the early access stage of a game. In addition, we expected that developers post more updates for an EAG, as they are improving the game (for example, based on the feedback that they acquire from user reviews).

Approach: We compared the average review rate ($\frac{\#\ of\ reviews}{\#\ of\ owners}$) within the early access stage and during the entire lifetime of a game. In addition, we compared the discussion

[6] http://store.steampowered.com/app/274230/

participation rate ($\frac{\#\ of\ posts}{\#\ of\ owners}$) within and after leaving the early access stage. We used the average review rate and the discussion participation rate to capture the interaction between players and the Steam platform. Furthermore, we calculated the update frequency (*# of days between adjacent release notes*) within and after leaving the early access stage. We used the update frequency to capture the interaction between developers and the Steam platform.

Because of promotional actions on Steam, the number of owners of a game can decrease. For example, there is a type of promotion named “free weekends”, which temporarily offers certain games at no charge. Players who get a game on a “free weekend” would only own the game for a limited time. However, these temporary owners are able to review a game as well. Hence, we used the highest number of owners that is observed during the early access stage and the lifetime of a game for our analysis.

We use the Wilcoxon signed-rank test to compare the metrics within and after leaving the early access stage. In addition, we calculate Cliff's delta d (Long et al., 2003) effect size to quantify the difference in the distributions of the metrics (see Section 4.4).

Findings: **63% of the EAGs update more frequently in their early access stage.** The beanplot in Figure 5.7 shows the distribution of the update frequency during and after leaving the early access stage. A beanplot shows the density plots for two distributions side by side so that they can be easily compared. In general, developers update their game more frequently in the early access stage, with a median of 11 days between adjacent updates in the early access stage. The number of days between releases after leaving the early access stage increases to 15 days. The Wilcoxon signed-rank test shows that the difference between the two distributions in Figure 5.7 is significant (p-value = 5.833e-10) with a small effect size (Cliff's Delta = -0.207). We calculated that

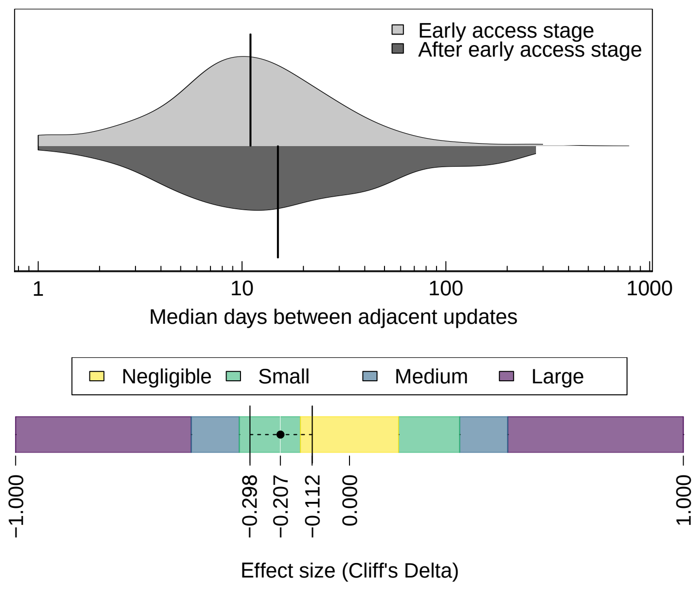

Figure 5.7: Distribution of the update frequency (measured as the median number of days between adjacent updates) during and after leaving the early access stage for all EAGs (the vertical line shows the median of each distribution). The figure below the beanplot shows Cliff's Delta effect size (-0.207) and its confidence interval ($[-0.298, -0.112]$). The colored areas represent the thresholds that we used to interpret Cliff's Delta.

almost two third (63%) of the EAGs have a higher update frequency in their early access stage, and 3% of the EAGs have the same update frequency in and after leaving their early access stage.

We inspected the update frequency of the EAGs that have a higher update frequency after their early access stage. 72% of these games have left the early access stage after 2015. A possible explanation for the update frequency being higher after leaving the early access stage is that the update frequency tends to be higher for a short-while directly after leaving the early access stage, because of a boost in new players or funds. After a while, the game tends to become more stable, resulting in a lower update frequency.

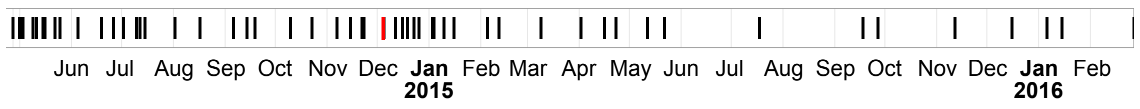

Figure 5.8: Update timeline of the *Fight The Dragon* game (each line represents an update, and the date of leaving the early access stage which is Dec 4, 2014 is marked in red)

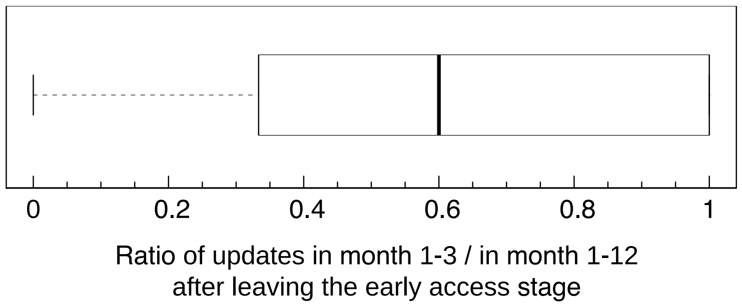

Figure 5.9: Distribution of the ratio of updates within 3 months and 12 months after leaving the early access stage.

Figure 5.8 shows an example of the update timeline of the *Fight The Dragon* game[7], a former EAG which has left the early access stage since December 2014. There exists a clear difference between the update frequency in and shortly after leaving the early access stage and the stage after June 2015.

To support our explanation of a stabilizing update frequency, we further studied the update timeline of former EAGs which have left the early access stage for at least a year. Figure 5.9 shows the distribution of the ratio of updates within 3 months and 12 months after leaving the early access stage. We calculated that for 29% of these former EAGs, 100% of their updates within 12 months after leaving the early access stage were released in the first three months. 51% of these former EAGs release 60% of the updates within the first three months.

[7]http://store.steampowered.com/app/250560/

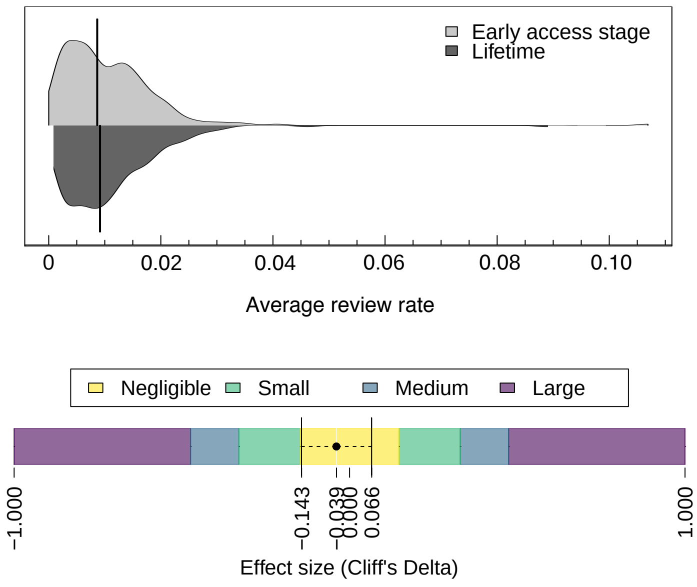

Figure 5.10: Distribution of the average review rate during the early access stage and the lifetime for all EAGs (the vertical line shows the median of each distribution). The figure below the beanplot shows Cliff's Delta effect size (-0.039) and its confidence interval ([-0.143, 0.066]). The colored areas represent the thresholds that we used to interpret Cliff's Delta.

65% of the EAGs see an equal or lower activity of owners posting reviews in the early access stage. Figure 5.10 shows the distribution of the average review rate during the early access stage and the lifetime. The Wilcoxon signed-rank test shows that the difference between the two distributions is significant (p-value = 0.009) with a negligible effect size (Cliff's Delta = -0.039), suggesting that users post reviews less often in the early access stage of a game than after leaving the early access stage. We calculated that 62% of the EAGs see an lower average review rate in the early access stage, and 3% of the EAGs see an equal average review rate in and after leaving the early access stage.

A possible explanation is that owners are aware that an EAG is still under development and not in its best shape. Hence, they prefer to give the developers more time to

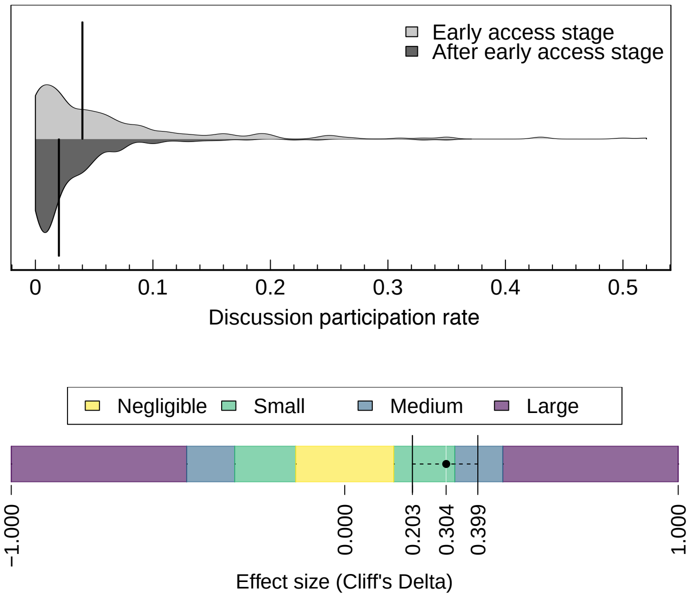

Figure 5.11: Distribution of the discussion participation rate during and after leaving the early access stage for all EAGs (the vertical line shows the median for each distribution). The figure below the beanplot shows Cliff's Delta effect size (0.304) and its confidence interval ([0.203, 0.399]). The colored areas represent the thresholds that we used to interpret Cliff's Delta.

improve the game, and wait until the game leaves the early access stage to give their reviews, rather than judge the game in its unfinished shape.

81% of the EAGs observe an equal or higher activity on the discussion forums in the early access stage. Figure 5.11 shows the distribution of the discussion participation rate during and after leaving the early access stage. As shown in the figure, a game receives a median of 0.04 discussion posts per owner in the early access stage, which is twice as high as the median number of discussion posts per owner after leaving the early access stage (0.02). The Wilcoxon signed-rank test shows that the difference between the two distributions is significant (p-value = 4.918e-16) with a small effect size (Cliff's Delta = 0.304). We calculated that 66% of the studied former EAGs observe a higher discussion participation rate in their early access stage. 15% of the studied former EAG have a consistent discussion participation rate in and after leaving the early access stage.

The higher discussion participation rate in the early access stage supports the explanation that we provided for the finding that owners post less reviews in the early access stage. It appears that early access owners tend to provide their feedback in discussion forums instead of in reviews, which does not affect the positive review rate of a game.

For developers, the lower review rate and the higher discussion participation rate in the early access stage appears to be a double-edged sword. On the one hand, the lower review rate reduces the chances that a possibly buggy and imperfect version of the game leads to complaints in reviews, which might mislead potential customers after leaving the early access stage. On the other hand, it is difficult for developers to perceive and quantify how satisfied the owners are in the early access stage. Although the discussion forums on the Steam Community offer a place for developers and players to communicate, the posts normally only consist of concrete issues such as questions or suggestions, rather than specific, quantifiable sentiment as provided by reviews.

Summary: Developers update a game more frequently in its early access stage. Players post significantly less reviews but more discussion posts in the early access stage (all with small effect size).

5.4.3 RQ3: How tolerant are players of the quality of EAGs?

Motivation: EAGs are unfinished by definition. Although owners have access to a playable version of a game, the content of this version can be incomplete, the client can be buggy or the performance can be poor. Players are aware of the possible issues when they purchase an EAG. Because the accumulated reputation during the early access stage can impact the popularity of a game after leaving the early access stage, we studied whether owners are more tolerant of the quality during the early access stage of a game.

Approach: We quantified the tolerance of owners of the quality of former EAGs within and after leaving the early access stage using the positive review rate of games ($\frac{\#\ of\ positive\ reviews}{\#\ of\ total\ reviews}$). The reviews of a game can greatly affect the will to purchase of potential customers. Hence, a higher positive review rate in the early access stage can lead to a higher popularity after the game leaves the early access stage. In addition, we calculated the correlation of the positive review rate, the length of the early access stage and the update frequency in the early access stage. We used Spearman correlation because our data is not normally distributed.

Findings: **89% of the EAGs receive an equally or higher positive review rate during their early access stage.** Figure 5.12 shows the distribution of the positive review rate during and after leaving the early access stage. The Wilcoxon signed-rank test shows that there is a significant difference (p-value < 2.2e-16) with a medium effect size (Cliff's Delta = 0.454) between the two distributions.

We calculated that 88% of the former EAGs receive a higher positive rate in their early access stage, with a median positive rate of 88%, which is higher than the median positive rate after leaving the early access stage (69%). 1% of the former EAGs receive a consistent positive rate in and after leaving their early access stage.

As mentioned in Chapter 2, the positive review rate is used in the games' Steam Store page as the official indicator of the quality of a game. As a result, a higher positive

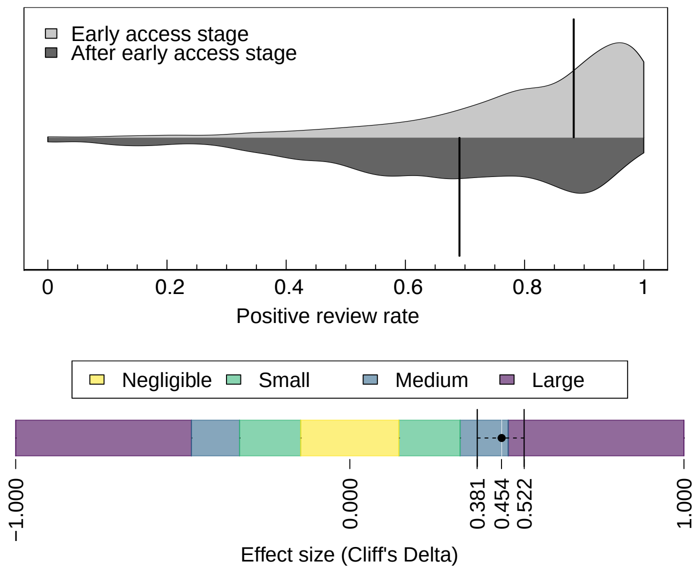

Figure 5.12: Distribution of the positive review rate during and after leaving the early access stage for all EAGs (the vertical line shows the median for each distribution). The figure below the beanplot shows Cliff's Delta effect size (0.454) and its confidence interval ([0.381, 0.522]). The colored areas represent the thresholds that we used to interpret Cliff's Delta.

review rate can greatly benefit the popularity of a game after it leaves the early access stage. The above findings suggest that games can receive more positive reviews when using the early access model. However, the higher positive review rate does not suggest that the early access model is a fix for low-quality games. More likely is the possibility that the people who buy EAGs are more tolerant of the unfinished status of a game. Another possibility is that the developers that use the early access model are good at keeping their players happy.

The positive review rate is not correlated with either the length of the early access stage or the update frequency in the early access stage The Spearman correlation between the positive review rate and the length of the early access stage is -0.06. The update frequency in the early access stage and the positive review rate have a Spearman correlation of 0.01. These findings indicate that neither the length of the early access stage, nor the update frequency in the early access stage are correlated with the positive review rate.

These findings suggest that developers can take time to polish their EAGs until they are ready to leave the early access stage, without worrying that the long length of the early access stage might decrease their positive review rate. In addition, developers can choose the update schedule that best fits their development process during the early access stage, rather than rush to add more content and features.

Summary: Players of EAGs tend to be more tolerant of the quality of a game during its early access stage. While players tend to post less reviews within the early access stage, 89% of EAGs receive an equally or more positive review rate in their early access stage. In addition, developers do not need to rush into releasing their games, as it appears that the tolerance of players is not correlated with the length of the early access, though other factors might be at play, such as the budget and funding of their games.

5.5 Additional Interesting Observations

In this section, we discuss several observations that are worth noting and can lead to future work.

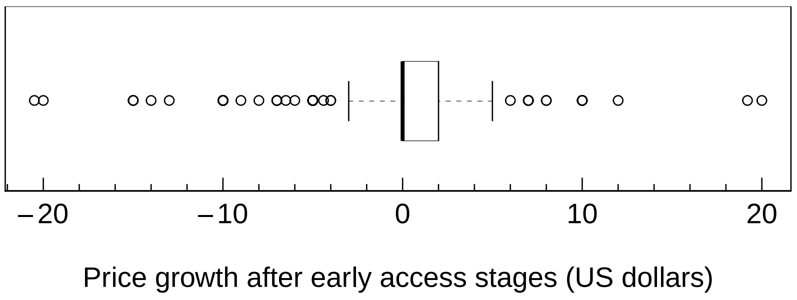

Figure 5.13: Distribution of price changes.

5.5.1 The Price of a Game Within and After Leaving the Early Access Stage

As explained in Section 5.2, developers of EAGs have the liberty to change the price of the game at any point in time. Steam states that, depending on the "goals and the level of commitment and feedback" developers desire from early access players, they can start by offering a discount, or on the contrary, charge a premium (Valve, 2016a). Therefore, we considered the change of price as a reflection of the purpose of developers to use the early access model. We assumed that developers ask a lower price in the early access stage when they aim to gather more feedback and use the low price to attract more players. On the other hand, when developers charge a higher price in the early access stage of their game, they tend to use the model as a funding source to support the development process of their games.

We compared the price during and after leaving the early access stage. Figure 5.13 shows the distribution of the price changes, i.e., we subtracted the early access price from the price of the game after leaving the early access stage. 95 (24%) of the EAGs are free to play throughout their lifetime, including the early access stage. We removed them from the figure for better demonstration. 145 (48.3%) of the remaining former

EAGs have the same median price within and after leaving the early access stage, while 91 (30.3%) increase their price and 64 (21.3%) decrease their price.

In addition, of the 64 games of which the price decreases, 6 (9%) become free to play after leaving the early access stage. We manually checked the release notes of these 6 games to identify the reasons for making the game free to play after leaving the early access stage. We were able to find the reasons for making the game free for three games, while the other three silently become free. When the *Pool Nation FX Lite* game[8] left the early access stage, developers divided the game into the basic free-to-play part and two optional packs which need to be purchased (CPx, 2015). The developers of the *Bierzerkers* game[9], however, stated that it is the early access players who suggested them to make the game free, in order to build the base of the game. To reward the early access players, they each received all of the launch characters (Bierzerkers, 2016). As for the *Cards and Castles* game[10], although developers did not specify the reasons of making the game free, they offered an early access bundle containing unique content to early access players and persuaded people to buy the game in the last two weeks of the early access stage (Cards and Castles, 2015).

For the free EAGs and the EAGs with a lower price in the early access stage, which represent 47% of the EAGs, it is likely that their developers focused on gathering early feedback from the community. The percentage is significantly higher than the EAGs which charge a premium for early access (16%), indicating that their developers aim at raising development funds. Although this is only one possible explanation for the change of price and developers might have several goals when using the early access

[8] http://store.steampowered.com/app/314000/

[9] http://store.steampowered.com/app/348460/

[10] http://store.steampowered.com/app/360730/

model, the phenomenon suggests that the majority of EAG developers value the opportunity to elicit feedback more than the opportunity to raise development funds.

5.5.2 Lessons Learned from an Early Access Failure: the *Spacebase DF-9* Game

In this section, we discuss lessons learned from the the *Spacebase DF-9* game. Prior work (e.g., Washburn Jr et al. (2016)) discussed what went wrong and what went right during the development of a game, but no prior work focused on the failure of EAGs specifically.

The *Spacebase DF-9* game is developed by *Double Fine Productions*, an indie game development studio (Fine, 2016). The game was available on the SEARP on October 15, 2013. On October 27, 2014, the game unexpectedly terminated the early access stage and released a final product that lacked many of the planned features. On November 21, 2014, twelve employers including the project lead were laid off. On December 16, 2014, an announcement was posted on the official technical support forums, stating that there were no further plans for patches and there was no team assigned to the project (Greg, 2014).

The abandonment of the game led to the disappointment of a large number of players. As a result, the game received 79% (2,598) negative reviews, and raised a debate between the players and the studio on the discussion forums of the game on the Steam Community (Valve, 2018).

The game is considered to be a failure of the early access model. In order to understand the reasons for its failure, and the lessons that can be learned for future EAGs, we manually studied two threads on the Steam Community. One of the threads is posted by the studio (TimOfLegend, 2014) and the other thread is posted by the players (ThunderPeel2001, 2015). Together the threads contain around 800 discussion posts. We identified the following lessons that can be learned:

Lesson 1: It is risky to use the early access model as the main funding source. The reason for terminating the development of the *Spacebase DF-9* game, claimed by the studio, is that this project was started with an open ended-production plan, with the hope that it can progress similarly to some other early access-funded games. However, the sales quickly became insufficient to support the development process. Although the developer put all the raised funds back into the development of the game, it turned out that the raised funds were not sufficient to fund a complete development team.

However, players argue that the developer should have considered the game as an investment, and that the profit would come after leaving the early access stage. They consider the funding of a game's development solely with early access sales to be *"irresponsible if not downright delusional"*. Some players even question the money management of the studio, although the studio later responded that it considered continuing development on a game that costs more than it makes to be bad money management.

Lesson 2: Do not release a game on the SEARP too early. A potential reason for not selling enough copies, posted by some players, is that the game was released into the SEARP too early, lacking content and features for players (*"basically nothing meaningful to do after 45 mins of playtime"*). The players suggested that, to ensure sales remain above made costs, developers should release the game in a more content and feature-rich state.

Lesson 3: State promises and plans clearly. The most obvious lesson that can be learned from the failure, as stated by the studio, is that it did not clearly indicate in the "original promise" which features were securely funded, and which portion of the game was dependent on early access sales. This point is supported by many players, who considered the original statement *"none of these features are set in stone"* to be too ambiguous and vague.

Lesson 4: When a game is abandoned by its development studio, the reputation of the studio as a whole can be damaged. Besides the anger towards the abandonment of the game, a large quantity of players are doubting the integrity of the studio, and claimed that they would never purchase any future game from *Double Fine Productions*. The players consider the abandonment of the EAG as betraying a long term commitment, as they purchased the EAG not for its current form, but for the potential it had. In addition, players were concerned about whether the studio would be capable of improving the development of future EAGs. The lack of introspection totally *"bankrupted the company by ruining the reputation"*, as said by players. It is worth noting that players stated clearly that they would not stop supporting EAGs or indie developers, but would specifically stop supporting this *"irresponsible"* studio.

Lesson 5: Communicate issues and changes to the promised plan on time. The studio claimed that they announced the situation and the decision to terminate the development rather than *"vanish quietly in the night"*. However, the players argued that if the studio could communicate with players immediately when trouble firstly came up, the players could have helped by recommending the game to friends and relatives, or even bought copies for them. It was the lack of communication of the troubles that the game was facing that killed this game.

The aforementioned lessons demonstrate that players get emotionally involved in the development of EAGs. One of the players posted that *“I am sorry that powers above you have closed your beloved project down, and I’m also frankly sorry that I don’t get a finished product. This game could have been so amazing.”*

Although the aforementioned lessons come from one game, they give an overview of realistic dangers that apply to EAGs. The main lesson that can be learned is that the player involvement should work in both directions. On the one hand, developers appreciate the feedback from the players of their game. On the other hand, developers should show appreciation of their players by communicating and actually involving them in the decision-making process.

5.6 Threats to Validity

This section presents the threats to the validity of our findings.

5.6.1 Internal Validity

A threat to the validity of our findings is that it is not necessary for game developers to publish release notes for a game update to one of the Steam channels. Hence, all numbers that we provide in this chapter may be low estimates of the actual number of updates.

The number of owners used in our study are estimated from a representative random sample by Steam Spy. Although a three-day rolling sample is used to increase the accuracy, there can still exist a deviation from the actual number of owners. However, because the sales data is confidential in the gaming industry, this is the most accurate method to our knowledge to estimate the number of owners of a game.

We estimated the total number of games that are released in a month using the release date as advertised on the Steam Store page. This number is an estimation because developers are allowed to change the release date that is shown on the Steam Store page. We observed that for some games that exist before they are released on Steam, developers changed the release date to the real release date. We do not have data (reviews, discussions, price, etc.) between the real release date and the date that the game is released on Steam. However, we expected it is sufficiently accurate to be used to give a reasonable estimate of the number of games released in a month. Note that we used release notes to identify the date on which a game was released as an EAG, hence this threat does not affect the validity of our findings that are related to EAGs.

The learned lessons that we described about failed games come from one game. However, at the time of writing, it is the only game for which the failure of the early access model has been explicitly documented. These lessons can be revisited later when documentation about the early access model for more EAGs becomes available.

As in all empirical studies, separating causation and correlation is a challenge in our work. While we cannot show that the early access model leads to more satisfied game owners, there exists a correlation between a higher positive review rate and the usage of the early access model. One possibility is that the type of game owner that buys an EAG is more happy in general than non-EAG buyers. Another possibility is that EAGs are only bought by more tolerant owners. Either possibility supports the findings that are presented in this chapter.

We used the frequency of game updates as a proxy of interaction between developers and the Steam platform. We believe that this interaction is a rough estimate of how much developers care about the quality of their game. In this chapter, we did not study whether updates are a direct response to user feedback. Future studies should investigate more closely the link between game updates and user feedback, for example, that is acquired through various avenues such as user reviews.

5.6.2 External Validity

In our empirical study, we studied the EAGs on Steam. The findings of our study may not generalize to other EAGs on different distribution platforms. However, as stated in Chapter 2, Steam is the largest digital distribution platform for PC gaming. Hence, the EAGs on Steam are representative for a large number of EAGs.

5.6.3 Construct Validity

We manually validated our approach for identifying release notes and observed that our approach has 89% precision and 87% recall, as described in Section 4.4.

5.7 Chapter Summary

In this chapter, we studied the early access release model for games. Games that are released through this model, so-called Early Access Games (EAGs), are early versions that allow developers to raise funds for development or to elicit early feedback from players. In particular, we studied the characteristics of 1,182 EAGs, the interaction between players and developers of EAGs and the Steam platform during and after leaving the early access stage, and the tolerance of players of the quality of EAGs. Below are the most notable findings of our study:

1. 15% of the games that are currently on Steam make use of the early access model. The most popular EAG has approximately 29 million owners.

2. EAGs tend to be “indie” games, which suggests that the early access model is used mostly by smaller development studios.

3. The percentage of players that review a game during its early access stage is lower than the percentage of players that review a game after leaving the early access stage. However, the average rating of the reviews is much higher during the early access stage.

In addition, we discussed several learned lessons from the failure of an early access game. The main learned lesson from this failure is that the communication between the game developer and the players of the EAG is crucial. Players enjoy getting involved in the development of an early access game and they get emotionally involved in the decision-making about the game.

Based on the findings that are presented in this chapter, we suggested the following to developers that are considering the early access model for releasing their game:

1. If you have a small marketing budget, the early access model can help you to build a positive reputation, as players of EAGs tend to give more positive reviews. However, the early access model will not lead to more reviews to your game.

2. Although you can get a larger amount of concrete feedback in the discussion forum, that feedback is not explicitly linked to negative or positive feelings (as is

the case with reviews), making it more difficult to quantify the feelings of your players in general.

3. Be aware that using the early access model as your main funding source is a risky strategy.

We believe that the findings of this chapter provide a first step in helping developers better understand the pros and cons of the early access model.

While our findings do not suggest that using the early access model inherently leads to more satisfied players, there exists a correlation between EAGs and a higher positive review rate. One possible explanation for this correlation is that players who buy EAGs are friendlier towards developers. Another explanation is that developers that use the early access model are good at keeping their players satisfied. Either way, while the early access model is not a fix for low-quality games, the early access model appears to be a valuable tool for developers that want to improve their games by interacting with their players.

Future studies should use methods such as developer surveys, user studies, and controlled experiments to examine in more depth the causality between using the early access model and the satisfaction of both players and developers.

In the following chapter we conduct an in-depth study of game reviews on the Steam platform.

CHAPTER 6

Studying Game Reviews on the Steam Platform

In Chapter 5, we studied early access games on the Steam platform, and observed that gamers provide more positive reviews to early access games. In this chapter, we perform an empirical study of the reviews of 6,224 games on the Steam platform, to better understand if game reviews share similar characteristics with mobile app reviews, and thereby understand whether the conclusions and tools from mobile app review studies can be leveraged by game developers. In addition, new insights from game reviews could possibly open up new research directions for research of mobile app reviews. We find that game reviews are different from mobile app reviews along several aspects. Additionally, the number of playing hours before posting a review is a unique and helpful attribute for developers that is not found in mobile app reviews. Future longitudinal studies should be conducted to help developers and researchers leverage this information. Although negative reviews contain more valuable information about the negative aspects of the game, such as mentioned complaints and bug reports, developers and researchers should also not ignore the potentially useful information in positive reviews. Our study on game reviews serves as a starting point for other game review researchers, and suggests that prior studies on mobile app reviews may need to be revisited.

An earlier version of this chapter is published in the Empirical Software Engineering Journal (EMSE) (Lin et al., 2018b).

6.1 Introduction

SIMILAR to mobile app distribution platforms, such as the Apple App Store and Google Play, many online game distribution platforms allow users to post reviews of a game. These game reviews provide a rich data source that can be leveraged to better understand user-reported issues. Prior work on mobile app reviews has shown the value of studying reviews (Vasa et al., 2012; Hoon et al., 2012; Khalid et al., 2015; Pagano and Maalej, 2013).

To get a deeper insight on the user-reported issues of games, in this chapter we study the reviews of 6,224 games on the Steam platform. As the first work that studies game reviews from a software engineering perspective, our goal is to understand if game reviews share similar characteristics with mobile app reviews. This understanding will allow us to reason about whether the conclusions and tools from prior mobile app review studies can be leveraged by game developers, thereby helping game developers understand better how to better leverage user reviews for improving the user-perceived quality of their games. In addition, our study could serve as a starting point for more longitudinal studies of game reviews, and possibly open up new research directions for research of mobile app reviews.

In the first part of this chapter, we conducted a preliminary study on the number, length, language and readability of game reviews. In addition, we studied whether there are game-specific characteristics that have a relation with the number of daily reviews. Our preliminary study shows that most games receive a limited number of reviews each day, with a relatively short length and high readability. There are several different aspects between game reviews and mobile app reviews. In addition, we observe that developers should be prepared to get a peak in the number of received reviews after a sales event.

In the second part of this chapter, we first studied what gamers talk about in their reviews, to understand if gamers address different things in their reviews than mobile app users. Second, we studied how long players play a game before they post a review. This information is unique compared to mobile app reviews, and may provide interesting insights for researchers to help developers design the storyline and levels of a game. In particular, we addressed the following two research questions (RQs):

RQ1: What are gamers talking about in reviews? We manually identified six categories of reviews. Although negative reviews contain more valuable information for developers, the portion of useful information in positive reviews, such as suggestions for further improving the game design, also should not be ignored by developers. Players appear to value game design over software quality (i.e., the number of bugs in a game).

RQ2: How long do players play a game before posting a review? Gamers play a game for a median of 13.5 hours before posting a review. The first hour playing experience is more important for free-to-play games, as we observed a peak in the number of received reviews for free-to-play games after approximately one hour of playing. Developers should pay particular attention to the design of the first 7 hours of gameplay, as the majority of negative reviews are posted within that period.

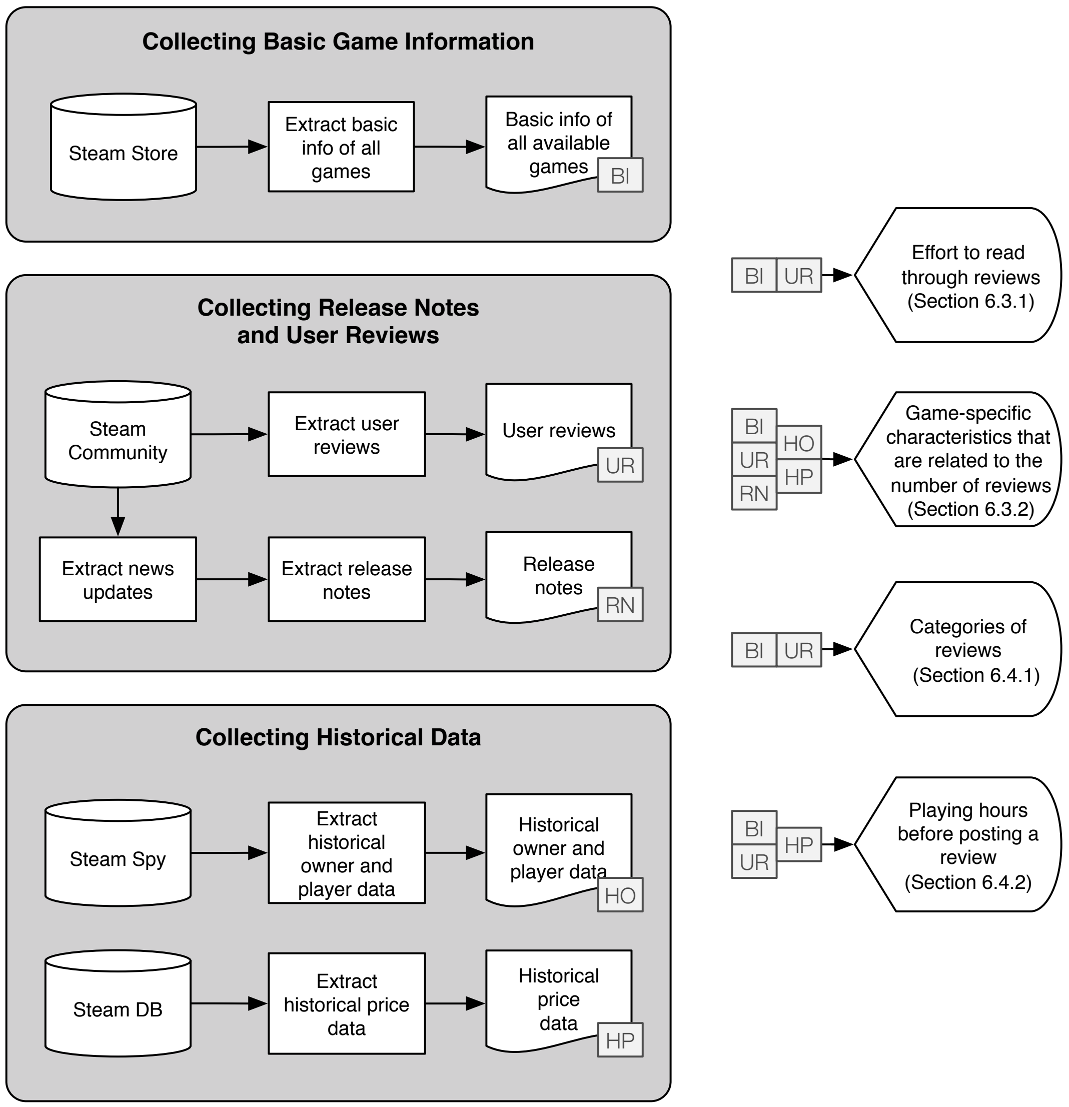
Collecting Basic Game Information
Steam Store
Extract basic info of all games
Basic info of all available games
BI
Collecting Release Notes and User Reviews
Steam Community
Extract user reviews
User reviews
UR
Extract news updates
Extract release notes
Release notes
RN
Collecting Historical Data
Steam Spy
Extract historical owner and player data
Historical owner and player data
HO
Steam DB
Extract historical price data
Historical price data
HP
BI UR
Effort to read through reviews (Section 6.3.1)
BI UR RN HO HP
Game-specific characteristics that are related to the number of reviews (Section 6.3.2)
BI UR
Categories of reviews (Section 6.4.1)
BI UR HP
Playing hours before posting a review (Section 6.4.2)

Figure 6.1: Overview of our study of game reviews

6.2 Our Methodology for Studying Game Reviews

This section introduces the methodology of our empirical study of game reviews. We detail how we extracted and processed data. Table 6.1 presents the description of our collected dataset. Figure 6.1 gives an overview of our methodology.

Table 6.1: Dataset description of our study of game reviews

# of studied games	6,224
# of news updates	98,823
# of release notes	37,613
# of reviews	10,954,956
# of positive reviews	9,393,546
# of negative reviews	1,561,410
# of English reviews	6,768,768
# of reviews with accurate playing hours data	28,159

6.2.1 Collecting Basic Game Information

We took a snapshot of all the 8,025 games that were available in the Steam Store on March 7th, 2016 using a customized crawler. It is important to select high-quality subjects when conducting software engineering research (Nagappan et al., 2013). As a result, prior studies on digital distribution platforms, such as mobile app stores, removed apps that do not have enough downloads as these apps are likely to be toy or personal projects. We removed games that had less than 25 reviews from our study, to avoid a possible bias in our results due to a small number of reviews. In total, we studied 6,224 games. We collected the title, developer, publisher, tags, genres, and current early access status (i.e. whether a game is in the early access stage or not) of games. The tags of a game are specified by its players, while the genres of a game are specified by its developer.

6.2.2 Collecting Release Notes and User Reviews

In order to obtain the update dates of games, we used the accompanying release notes that are posted on channels in the Steam Community. We used the process described in Section 4.3 to extract release notes from the channels. We identified 37,613 release notes for the studied 6,224 games.

We extracted all the reviews for each game from the Steam Community, and filtered-out reviews that contain no words, but only random characters such as smiley faces (e.g., ":)"), as they are non-informative and can be easily filtered-out by developers. There were in total 10,954,956 reviews across all supported natural languages. Steam provides a filter for the language of reviews for a game. We crawled the reviews in each language separately using this filter, to identify the language of each review. However, the number of playing hours (i.e., the number of hours that the reviewer played the game) that is shown with each review is not the number of playing hours at the time of posting the review, but the number of playing hours until now. Hence, in order to study the timing of gamers posting reviews, we developed another real-time crawler which only crawls reviews that are received within the last 6 minutes of the time of crawling, to collect reviews that have an accurate number of playing hours. Therefore, we were able to collect the dataset with an error margin of 6 minutes. We ran the real-time crawler for a month and collected 28,159 reviews with an accurate number of playing hours.

6.2.3 Collecting Historical Data

We collected the history of the number of owners, the number of players, and the price for all studied games using the approach described in Section 5.3.

6.2.4 Types of Studied Reviews

We used the developer-provided game genres to distinguish two types of games. We considered games that are tagged with the "Indie" genre as indie games, and games that are tagged with the "Free-to-play" genre as free-to-play games. In addition, we identified early-access games using the crawled data (see Section 6.2.1). In our study, we compared all the studied reviews along the following four dimensions:

1. **Positive reviews and negative reviews.** Prior work has shown that positive reviews and negative mobile app reviews may provide different information (Pagano and Maalej, 2013). We studied whether positive and negative game reviews are different from each other as well.

2. **Indie game reviews and non-indie game reviews.** Because of the rise of digital distribution platforms such as the Steam platform, indie games have become an important part of the gaming industry after 2004, as these platforms offer a convenient way of distributing games from studios with a smaller budget (Cobbett, 2017). To the best of our knowledge, there is no official definition of "indie" games. We use the universal definition as proposed by Stern (2012): *"A game that is both (a) developed to completion without any publisher or licensor interference, and (b) created by a single developer or a small team."* We assume that the "indie" genre on Steam follows this definition. As the team size and available development resources are very different between indie games and non-indie games, we studied indie games and non-indie games separately from each other.

3. **Early access reviews and non-early access reviews.** Our earlier analysis has shown that players of early access games interact differently with the Steam

platform during the early access stage (see Chapter 5). Hence, we studied if early access reviews are different from non-early access reviews.

4. **Free-to-play game reviews and non-free-to-play game reviews.** We explored if paying for a game has an impact on a user's review behavior.

For each dimension, we compared the total number of reviews, and we manually studied exceptional cases or extraordinary findings. Note that a review can fall into several dimensions (e.g., a review of an indie game can also be a review of a free-to-play game).

6.3 Preliminary Study of the Characteristics of Game Reviews

In this section, we present our preliminary study of the characteristics of game reviews. As shown in prior work (Vasa et al., 2012; Hoon et al., 2012; Pagano and Maalej, 2013; Khalid et al., 2015; McIlroy et al., 2017), mobile app reviews contain useful information for developers to improve the quality of the apps. Similarly, we expect that game reviews will contain valuable information for game developers. It is obvious that the best solution for understanding the issues that users raise in reviews, is to manually read through all the reviews. However, popular games may receive a large number of reviews each day, making it time-consuming for developers to read through all of them. Moreover, the number of reviews that are received each day is under constant fluctuation. For example, Figure 6.2 shows the daily number of reviews of the *Dota 2* game. The figure suggests that the number of reviews that are received each day is

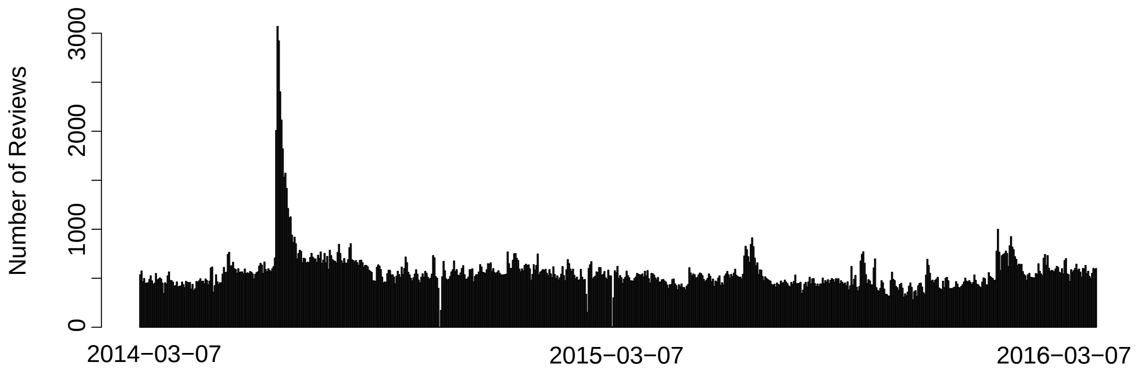

Figure 6.2: The number of reviews that the *Dota 2* game received each day. On July 19, 2014 (the peak in the figure), Steam ran a large scale sales event named "Steam Summer Sale 2014", during which players of the game could win free premium game items.

under constant fluctuation, making it hard for developers to assign resources to read through reviews.

In our preliminary study, we first studied the number of reviews that games receive each day, the length of the reviews, and the readability of reviews, to understand if game reviews share similar characteristics with mobile app reviews. We then investigated the impact of different game-specific characteristics on the number of reviews that are received each day, to understand what drives this number, and whether the phenomenon is consistent with that of mobile app reviews. Such analysis would answer the question of whether game developers can directly adopt conclusions from prior work on mobile app reviews, and whether prior work on automatically extracting useful information from mobile app reviews can be directly applied to game reviews. Table 6.2 shows the description of the dataset that is used in this preliminary study along four of the studied dimensions from Section 6.2.

Table 6.2: Dataset description of our preliminary study of the characteristics of game reviews

	Indie	**Non -indie**	**Early access**	**Non-early access**	**Free -to-play**	**Non-free -to-play**
# of games	3,628	2,596	552	5,672	384	5,840
# of positive reviews	3,664,191 (86%)	5,729,355 (86%)	973,191 (81%)	8,420,355 (86%)	1,784,118 (84%)	7,609,428 (86%)
# of negative reviews	601,376 (14%)	960,034 (14%)	229,125 (19%)	1,332,285 (14%)	338,729 (16%)	1,222,681 (14%)
# of all reviews	4,265,567	6,689,389	1,202,316	9,752,640	2,122,847	8,832,109

6.3.1 How many reviews are posted and what is their complexity?

Approach: We studied the number and the complexity of reviews from three perspectives: the number of reviews to read each day, the length of the reviews, and the readability of reviews. We studied the readability of all 6,768,768 English reviews, and the number and length of all 10,954,956 collected reviews.

In order to compare the scale and the complexity of different types of reviews, we used the Wilcoxon signed-rank test (Wilcoxon, 1945) (see Section 4.4) to compare the distributions for the metrics of different groups of reviews. We grouped the reviews by several different aspects including positive versus negative, early access versus non-early access, free to play versus non-free to play, and different genres. For example, we calculated the medium length of positive reviews and negative reviews of the *Counter-Strike* game, which is 18 characters and 45 characters respectively. We considered the medium length of positive and negative reviews of a game as a pair as the reviews come from the same group of players. We repeated the process for all the studied games, then applied the Wilcoxon signed-rank test to all the pairs. We used a paired test in this section because players of different games may have different review habits, hence by

Table 6.3: Examples of reviews with low and high Coleman-Liau Index (CLI)

Example	**Review content**	**CLI**
A review with a low CLI	*"Very good game, but it was not as good as the first one. It's a fun little game to pass your time, and it's FREE."*	3.7
A review with a high CLI	*"Of course, ironically in the exact same way as robocraft met it's downfall, it was ruined by Greedy developers trying to force their player-base to spend money on microtransactions with anti-consumer methods of getting the weapon parts they actually want."*	14.4

using a paired test we ensured that we were comparing different types of reviews for the same game.

In addition, we calculated Cliff's delta d (Long et al., 2003) effect size to quantify the difference in the distributions of the metrics (see Section 4.4).

We quantified the readability of reviews using the Coleman-Liau index (Coleman and Liau, 1975). The Coleman-Liau index is a readability test that is designed to gauge the understandability of a piece of text. The index approximates the U.S. grade level thought necessary to comprehend the text. Unlike other readability tests (e.g., the Flesch reading index (Kincaid, 1975)), The Coleman-Liau index (CLI) avoids the problem of inaccurately counting syllables (Coleman and Liau, 1975). The CLI is calculated using the following formula:

$$CLI = 0.0588L - 0.296S - 15.8$$

where L is the average number of letters per 100 words, and S is the average number of sentences per 100 words. A higher CLI indicates that the text is harder to understand,

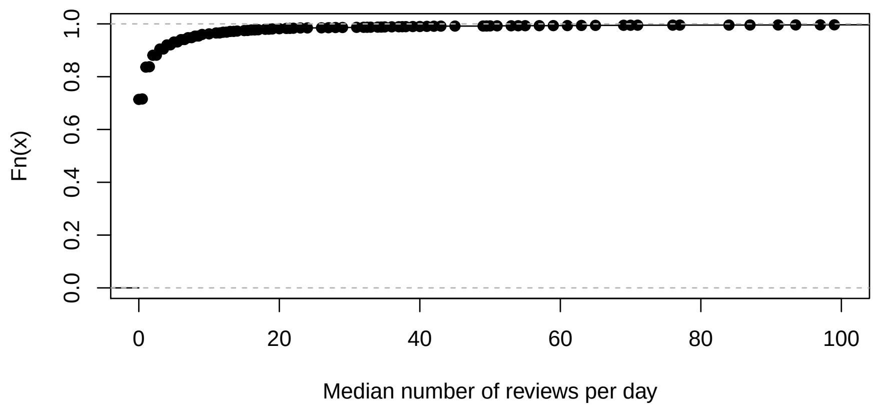

Figure 6.3: The empirical cumulative distribution function (Fn(x)) of the median number of reviews that are received by each game per day

while a lower CLI indicates that the text is easier to understand. Hence, reviews with a lower CLI should be easier to read through. Table 6.3 shows examples of reviews with a low and a high CLI respectively.

Findings: **96% of the games receive a median of less than 10 reviews per day.** Figure 6.3 shows the empirical cumulative distribution function of the median number of reviews that are received by each game per day. We removed 20 games with a median number of reviews per day that is greater than 100 from the figure for better demonstration. As shown in Figure 6.3, the distribution is extremely skewed. On average, a game receives a median of 2 reviews per day, and 96% of the games receive a median of less than 10 reviews per day. As a result, it should be practical for developers of most games to manually go through all received reviews. However, even developers of games with a relatively low number of reviews per day may not be able to go through all reviews. For example, developers of indie games may only have limited time each week to spend on a game (e.g. because they have an additional full-time job). Hence, the number of reviews that needs to be read for those games could still add up fairly quickly.

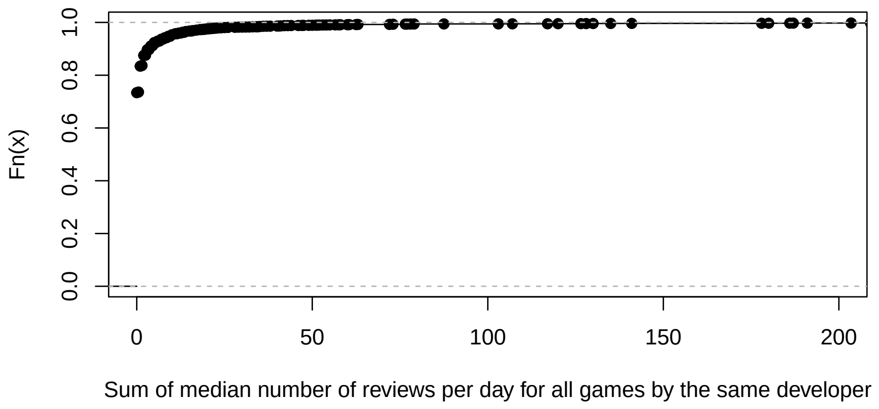

Figure 6.4: The empirical cumulative distribution function (Fn(x)) of the sum of median numbers of reviews per day for all games developed by the same developer

It is worth noting that the number is lower than the number of reviews received by mobile apps, which is a median of 22 reviews per day per mobile app. In particular, mobile app users published a median of 31 reviews per app per day for apps in the Games category (Pagano and Maalej, 2013).

We manually examined the games that received a median of more than 100 reviews each day, and observed that these games are either recently released games, or very popular games. For instance, the *Stardew Valley* game[1] was released on February 26, 2016, less than two weeks before our data collection, and received a median number of 586 reviews per day, which is the highest in our dataset. On the other hand, after being released more than 3 years ago, the *Counter-Strike: Global Offensive* game[2], which has more than 25 million owners, receives a median of 514.5 reviews per day. The observation suggests that there may exist game-specific characteristics that have a relation with the number of reviews received by games, such as the lifetime and the number of

[1]http://store.steampowered.com/app/413150
[2]http://store.steampowered.com/app/730

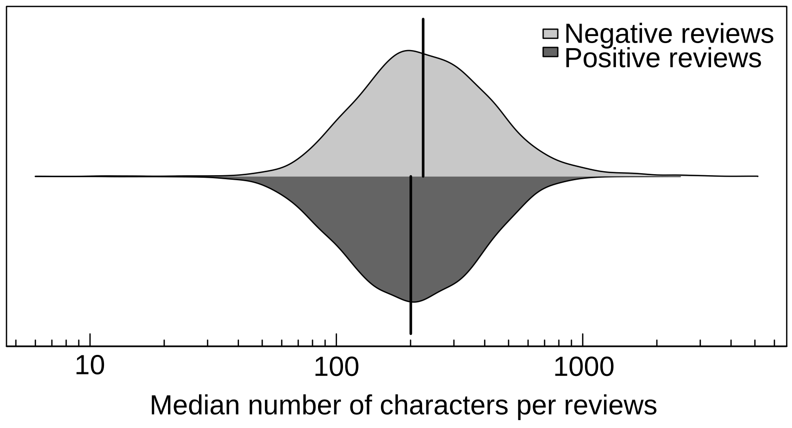

Figure 6.5: The distribution of the median length of positive and negative reviews per game. The vertical lines represent the median. The distributions are significantly different ($p < 0.05$), with negligible effect size.

owners of the games. We further study the relation of such game-specific characteristics with the number of reviews that are received by games per day in Section 6.3.2.

In addition, we grouped the games by developer, to study how many reviews per day a developer of multiple games would potentially need to read. For this calculation, we included games with less than 25 reviews as well, to get a more accurate overview of the total number of reviews for a developer. Figure 6.4 shows the empirical cumulative distribution function of the sum of median numbers of reviews that were received by all games from the same developer per day. As shown in the figure, 99% of the developers receive less than 50 reviews in total from all their games.

Most games receive reviews with a median length of 205 characters, or 30 words. Figure 6.5 shows the distribution of the median length of reviews. We calculated that the median value of the median number of words in reviews per game is 30 words. The lowest median length of reviews is 15 characters for reviews of the *Karos Returns* game[3]

[3]http://store.steampowered.com/app/371310

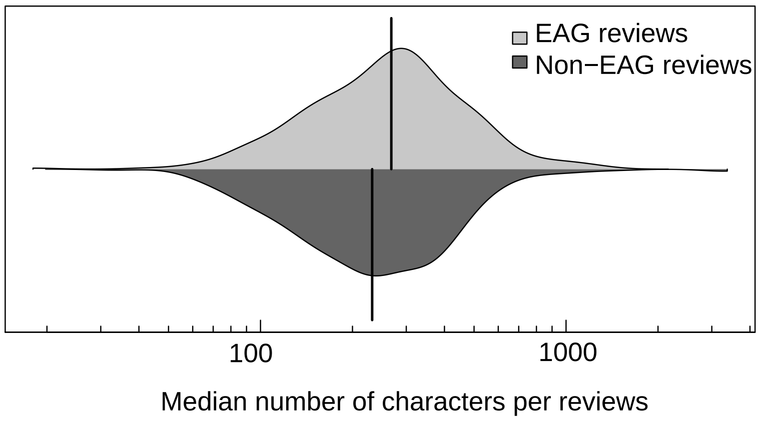

Figure 6.6: The distribution of the median length of early access reviews and non-early access reviews per game. The vertical lines represent the median. The distributions are significantly different ($p < 0.05$), with a small effect size.

(245 reviews in total), and the highest median length of reviews is 1,684 characters for reviews of the *Drizzlepath: Genie* game[4] (29 reviews in total).

We calculated that the median length of reviews across all the games is 93 characters. Our findings show that game reviews are longer than mobile app reviews, which have a median of 61 characters (Pagano and Maalej, 2013).

Negative reviews are slightly longer than positive reviews, but the difference is negligible. Figure 6.5 shows the distribution of the median length of negative and positive reviews. The Wilcoxon signed-rank test shows that the two distributions are significantly different, however with a negligible Cliff's delta effect size. The negligible effect size indicates that although negative reviews are slightly longer in general, the difference is negligible.

Early access reviews are slightly longer than non-early access reviews. Early access reviews are reviews that are received in the early access stage of an early access

[4]http://store.steampowered.com/app/438340

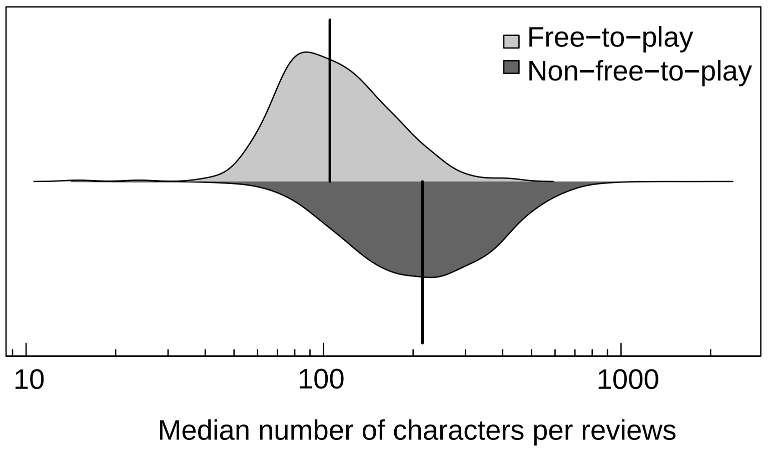

Figure 6.7: The distribution of the median length of reviews for free-to-play and non-free-to-play games. The vertical lines represent the median. The distributions are significantly different ($p < 0.05$), with a large effect size.

game[5]. Early access games allow players to purchase the game during its public beta period while developers continue working on the game. Developers of early access games can receive crucial feedback and bug reports directly from their target community in an earlier development phase. Hence, players may provide more detailed feedback in early access reviews. Our earlier analysis (see Chapter 5) showed that the average rating of reviews is higher during the early access stage. Figure 6.6 shows the distribution of the median length of early access and non-early access reviews. The Wilcoxon signed-rank test shows that the two distributions are significantly different, with a negligible effect size, indicating that early access reviews are slightly longer than non-early access reviews.

Players write longer reviews for games for which they paid. Figure 6.7 shows the distribution of the median length of reviews for free-to-play and non-free-to-play games. Free-to-play game reviews have a median length of 105 characters per game,

[5]http://store.steampowered.com/earlyaccessfaq/

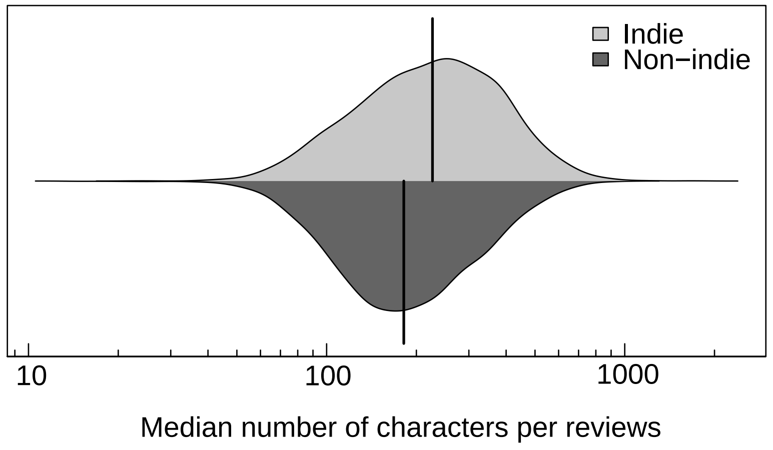

Figure 6.8: The distribution of the median length of reviews for indie and non-indie games. The vertical lines represent the median. The distributions are significantly different ($p < 0.05$), with a small effect size.

while non-free-to-play games have a median length of 215 characters per game. The Wilcoxon signed-rank test shows that the two distributions are significantly different, with a large Cliff's delta effect size, indicating that non-free-to-play games receive longer reviews than free-to-play games. One possible explanation is that paying for a game makes players feel more strongly about that game.

Reviews for indie games are longer than reviews for non-indie games. Figure 6.8 shows the distribution of the median length of reviews for indie and non-indie games. The Wilcoxon signed-rank test shows that the two distributions are significantly different, with a small Cliff's delta effect size. The difference is similar to the length of reviews for early access games and non-early access games. The similarity could possibly be explained by the fact that early access games are mostly indie games, as shown in our earlier analysis (see Chapter 5).

Games receive a median of 36% non-English reviews. Figure 6.9 shows the distribution of the portion of reviews in the top 10 languages. We studied the games with

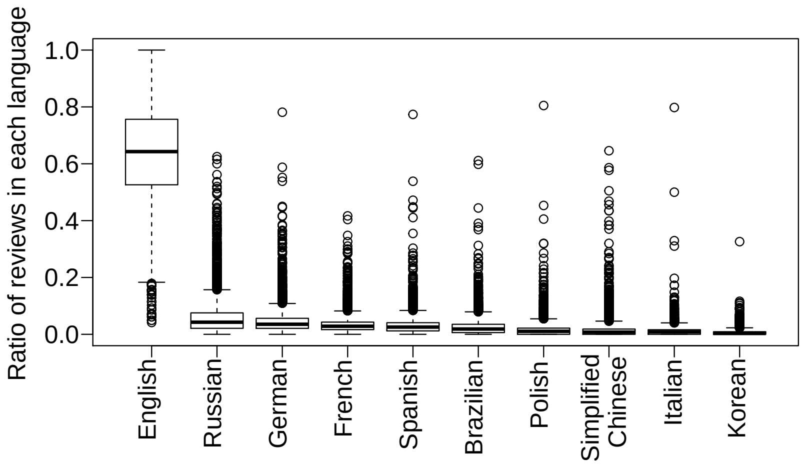

Figure 6.9: The distribution of the portion of reviews in the top 10 languages per game

a low portion of English reviews, and observed that most of them were developed by studios from non-English speaking countries. Although some of these games have an English interface, the majority of their customers may not speak English. In comparison with mobile app store research, which is usually done on the U.S. version of a store, review language poses a larger threat on Steam, as there is only a single global Steam store. Hence, future studies need to be aware of the considerably large portion of non-English reviews.

Reviews have a median readability level of grade 8. Figure 6.10 shows the distribution of the median Coleman-Liau index (CLI) of reviews in English. The median value of the distribution is 7.83, and the first and the third quartiles of the distribution are 7.20 and 8.43 respectively, indicating that most game reviews have a median readability level of around US grade 8. We did not observe significant differences in the CLI distribution across different genres of games.

We calculated the median Coleman-Liau index for the reviews of each mobile app in the dataset provided by Grano et al. (2017). The reviews of mobile apps in the dataset

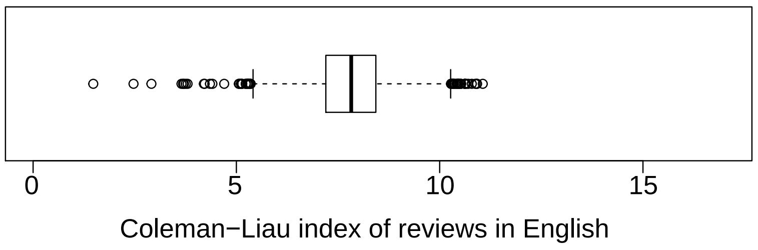

Figure 6.10: The distribution of the median Coleman-liau index of reviews in English.

have a median CLI of 5.69, which is lower than the median CLI calculated for game reviews. The Wilcoxon rank sum test confirms the significant difference between the readability of game reviews and mobile app reviews, with a large effect size. Game reviews have a significant lower readability than mobile app reviews.

Summary: Most games receive a limited number of reviews each day, with a relatively short length and high readability. Reviews of early access games are slightly longer. More advanced review selection and summarization techniques are needed for developers of the top 4% games with the most reviews, or for developers who cannot go through their daily reviews for other reasons.

6.3.2 Which game-specific characteristics are related to the number of reviews that are received each day by a game?

Approach: We investigated what drives the number of reviews, and whether the phenomenon is consistent with that of mobile app reviews, so that developers can better assign resources to deal with a sudden growth in the number of reviews. We investigated the impact of different game-specific characteristics on the number of reviews that are received each day, including the age of the game, the number of players and the

Table 6.4: A description of the variables of the mixed-effect model

Dependent variables	**Effect type**	**Type**	**Description**
game_id	Random	Categorical	The Steam game id.
developer	Random	Categorical	The developer of the game.
studio_size	Random	Numeric	The number of games that are developed by the developer.
owners	Fixed	Numeric	The number of owners of the game on that day.
players	Fixed	Numeric	The number of players of the game on that day.
eag	Fixed	Boolean	Whether the game is in the early access stage.
age	Fixed	Numeric	The number of days since the initial release of the game.
last_update	Fixed	Numeric	The number of days since the last update of the game.
last_discount	Fixed	Numeric	The discount percentage of the last sale.
last_discount_life	Fixed	Numeric	The number of days after the last sale.

number of owners[6], the developer, the size of the developer studio, information about discounts, and the number of updates. We studied the impact of the aforementioned game-specific characteristics on the number of reviews by building a linear mixed-effect model (Bates et al., 2015), using all 6,224 studied games as training dataset. In a traditional linear regression model, all the independent variables have the same relation with the dependent variable, hence such a model cannot express differences for independent variables at different hierarchical levels (e.g., different games). Unlike traditional linear regression models, linear mixed-effect models have two types of variables, i.e., random effect variables (game-level variables) and fixed effect variables (review-level variables). A mixed-effect model expresses the relationship between the dependent variable (i.e., the number of reviews that are received in one day for a game) and the review-level variables (e.g., the number of players of the game on that day),

[6]Anyone who purchased the game is the owner of the game, but only the people who played the game on that day are counted as the player of the game.

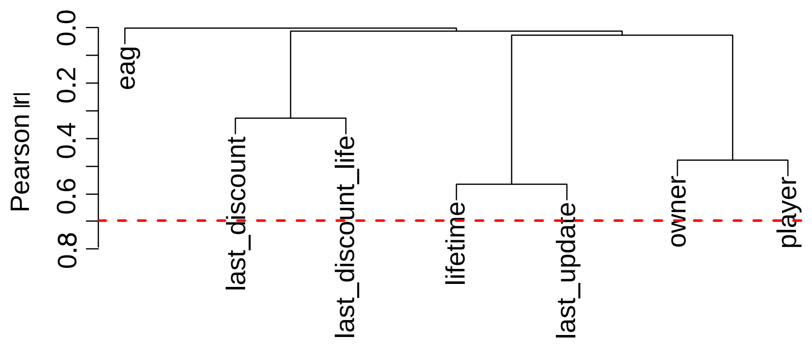

Figure 6.11: Hierarchical overview of the correlation among the fixed effect variables. The dotted line shows the threshold ($|\rho| = 0.7$)

while taking into consideration the different game-level metrics (e.g., the Steam game id). Table 6.4 shows the independent variables that were used in the model.

Data Scaling. Prior to building our model, we centered and scaled the data, so that we can interpret the coefficients in the model. We applied the `scale` function in R to the numeric variables in Table 6.4.

Correlation Analysis. We checked for fixed effect variables that are highly correlated with one another using Pearson correlation. We used a variable clustering analysis to construct a hierarchical overview of the correlation among the fixed effect variables. We selected only one variable from the sub-hierarchy with correlation $|\rho| > 0.7$ (McIntosh et al., 2016) for inclusion in our models. Figure 6.11 shows the hierarchical overview of the correlation. There is no fixed effect variable over the threshold. Hence, all variables are kept for further analysis.

Redundancy Analysis. Redundant variables (i.e., variables can be explained using other explanatory variables) in an explanatory model will distort the modelled relationship between the explanatory and dependent variables. We used the `redun` function in the `rms` package to detect redundant variables. With a threshold for $R^2 = 0.9$,

Table 6.5: Model result for fixed effects

	Coefficient	Standard Error	p-value	Significant
players	72.77	0.30	$< 2e-16$	✓
lifetime	-4.27	0.33	$< 2e-16$	✓
owners	3.91	0.16	$< 2e-16$	✓
last_discount	2.62	0.07	$< 2e-16$	✓
eag	2.37	5.22	0.650163	
last_update	0.83	0.22	0.000231	✓
last_discount_life	0.29	0.08	0.000496	✓

Table 6.6: Model result for random effects

Groups	Variance
game_id	132.030
studio_size	14.102
developer	0.426

all variables survived the redundancy analysis, and hence were kept for building the model.

Model Building. We used the `lmer` function in the `lme4` package to build the linear mixed-effect model. We also used the `lmerTest` package to calculate the p-value for each fixed effect variable. Table 6.5 and Table 6.6 shows the result of the model.

Findings: **The number of players has the strongest relation with the number of reviews.** Table 6.5 shows the mixed-effect model results for fixed effects. The “players” variable has the highest estimated coefficient, while the “owners” variable’s estimated coefficient is not high, indicating that the number of active players has a stronger relation with the number of reviews received each day than the owner base.

Although the finding that the number of players has the strongest relation with the number of reviews may look trivial, it actually yields us new information compared to prior studies, such as those of mobile app reviews. Such studies only had access to the number of owners of an app, while our study has access to both the number of owners and the number of active users (players).

A sale event has a stronger relation with an increase in the number of received reviews than releasing an update. It is also worth noting that in Table 6.5, the “last_discount” variable has the third highest estimated coefficient, while the “last_update” variable has the second lowest absolute estimated coefficient, indicating that a sale event has a greater impact on increasing the number of reviews than releasing an update. A possible explanation is that a sale event can increase the number of players, leading to a higher number of reviews.

The finding is not consistent with prior studies of mobile app reviews. Prior studies on mobile apps tend to ignore paid apps altogether (although there are exceptions (Maalej and Nabil, 2015; Fu et al., 2013)). Our finding shows that discounts are an important factor when studying reviews. In addition, prior studies of mobile apps have shown that mobile app reviews are generally triggered by new releases (Pagano and Maalej, 2013). However, our study shows that of all seven fixed effects that we considered, the number of days since the last update of the game has the second lowest impact on the number of received reviews. Hence, our results are an indication that prior mobile app studies may need to be revisited, thereby taking discounts in the app store into account as well.

Summary: A sale event has a stronger relation with an increased number of reviews than releasing an update. Developers should be prepared to get a surge in the number of reviews after a sales event.

6.4 Results of our Study of Game Reviews

In this section, we present the results of our empirical study of game reviews on the Steam platform. First, we discuss the categories of game reviews, and compare those to the taxonomy of mobile app reviews from prior work, to understand if gamers address different things in their reviews than mobile app users. Then, we study the number of playing hours before posting a review. As the number of playing hours before posting a review is a very unique attribute of game reviews compared to mobile app reviews, we study whether this attribute provides interesting insights for researchers, e.g., to provide developers advice for designing the storyline and levels of a game. Our findings can also demonstrate the value of collecting and analyzing app usage times for mobile app developers and researchers.

6.4.1 RQ1: What are gamers talking about in reviews?

Motivation: In our preliminary study, we studied the reviews from several quantitative views. In this step, we completed our study of reviews using a qualitative approach. Our goal was to understand what are the differences between the content of game reviews and the content of mobile app reviews, and among different types of games. We classified reviews into high-level categories, and compared our findings to mobile app review studies.

Approach: We manually categorized a statistically representative random sample of English reviews. To obtain the sample, we followed the following steps:

1. To select a representative sample with a confidence level of 95% and a confidence interval of 10%, we need to randomly select at least 96 reviews for each studied

dimension (based on the total number of reviews in that dimension). Hence, we randomly selected reviews from the population of all English reviews, and we counted the number of selected reviews from each dimension, until we selected at least 96 reviews from each dimension.

2. We randomly selected 96 reviews from each dimension from the reviews that were selected in the first step, to create a sample of equal size for each dimension.

3. We ended up with a representative sample that contains 472 reviews in total, and 96 reviews across each studied dimension (note that a review may appear in multiple dimensions). When considering the representativeness of the sample of 472 reviews, we could draw conclusions with a 95% confidence level and 5% confidence interval. We could also draw conclusions at the dimension-level with a 95% confidence level and a 10% confidence interval.

We performed an iterative process that is similar to *Open Coding* for classifying reviews, as suggested by Seaman et al. (2008); Seaman (1999). The procedure is shown in Listing 6.1.

```
Inputs = All reviews, a list of categories of reviews (which is
    initially empty)

For each review:
   Manually examine the content of this review.

   If the review matches an existing category:
      Label the review with that/those category(-ies).

   Else:
      Add a new category to the list of categories of reviews.
      Restart labelling with new list of categories.

Outputs = All reviews (labelled with appropriate categories), and
    a list of categories of reviews
```

Listing 6.1: Coding process

The procedure starts with an empty list of categories of reviews. For each of the reviews in the sample set, we manually examine the content of the review. If the review matches one or more existing categories in the list, we label the review with those categories. Otherwise, we add a new category to the list and restart labelling with the new list of categories. Note that a review can be categorized into more than one category. For example, if a review contains a suggestion for the game as well as reports a bug, the review would be categorized into both the "Suggestion" and "Bug" categories. Two researchers including the author of the thesis and a collaborator performed the process individually, and compared the results. The two researchers had a (partially) different categorization for 85 out of 472 reviews. The vast majority of the disagreements were cases in which one of the coders assigned an additional label to a review. The conflicts were easily resolved by discussing and coming to an agreement.

Table 6.7: Identified categories of reviews of games on the Steam platform

Category	**Description**	**Example**
Not helpful	The review contains information that is not helpful for a developer, such as stating the emotion without giving specific reasons.	*"Good game!"*
Pro	The review contains a pro of the game.	*"This game does atmosphere well and the story presented a mystery that seemed worth exploring ..."*
Con	The review contains a con of the game (excluding a bug).	*"... a very, VERY, sharp learning curve ..."*
Video	The review contains an URL to a video review.	*"For my full review please vist: [YouTube link], And watch my video!"*
Suggestion	The review contains a suggestion on how to improve the game.	*"... The game would have been more interesting if you could play with 4 players ..."*
Bug	The review contains a description of a bug that occurs in the game.	*"My game crashed, not once, not twice, but three times in five minutes..."*

During our analysis, we extracted 6 categories from the reviews. Table 6.7 shows all categories with their description and an example taken from an examined review.

Several studies of mobile app reviews have proposed taxonomies for mobile app review contents (Pagano and Maalej, 2013; Gu and Kim, 2015; Di Sorbo et al., 2016; Ciurumelea et al., 2017). However, some of these studies focused on the intention instead of the content of reviews (Di Sorbo et al., 2016), while others were only applicable to mobile apps (Ciurumelea et al., 2017). As shown in the previous sections, there are differences between mobile app reviews and game reviews. Therefore, we did not follow the taxonomies proposed for mobile app reviews in this section. We compare our extracted categories to the high level mobile app categories that were proposed by Gu and Kim (2015) in Table 6.8.

Table 6.8: Mapping between Gu and Kim's categories (Gu and Kim, 2015) and the categories of game reviews that are identified in this chapter

This chapter	Gu and Kim (2015)
Not helpful	Praise, Others
Pro	Aspect Evaluation
Con	Aspect Evaluation
Video	Others
Suggestion	Feature Request
Bug	Bug Report

Table 6.9: Categories of reviews of games on the Steam platform (ordered by % of all reviews)

Category	% of all reviews	% of positive reviews	% of negative reviews	% of early access reviews	% of non-early access reviews	% of reviews for indie games	% of reviews for non-indie games	% of reviews for free-to-play games	% of reviews for non-free-to-play games
Not helpful	71	71	55	68	72	66	73	66	71
Pro	38	46	18	41	32	41	30	25	36
Con	34	29	57	27	33	40	31	30	33
Bug	8	7	17	13	8	7	9	14	7
Suggestion	4	4	2	9	2	3	1	3	4
Video	1	1	1	4	1	2	1	2	1

Note that these percentages do not add up to 100% as a single review can be labelled to multiple categories.

Findings: **42% of the reviews provide valuable information to developers.** Table 6.9 shows the percentage of each category. We calculated that the categories that provide valuable feedback for improving the games (i.e., "Pro", "Con", "Suggestion", "Bug") cover 42% of the reviews, suggesting that it is important for developers to read through reviews. The percentage is slightly higher than the 35% "informative" reviews (i.e., reviews that are potentially useful for developers to improve the quality of the user experience of apps) in mobile app reviews (Chen et al., 2014).

It is worth noting that, although some categories may not be valuable for developers (e.g., "Not helpful"), they may be helpful to other players or potential customers.

Players complain more about game design than bugs. Table 6.9 shows that only 8% of the reviews mention bugs in games, while 34% of the reviews mention the cons related to game design. Moreover, in negative reviews, 17% mention the bugs while 57% mention the cons related to game design. The percentage of reported bugs in reviews is surprisingly low compared to the cons of game design, suggesting that players value a well-designed gameplay over software quality (i.e., the number of bugs in a game).

Moreover, 42% of the reviews that mention bugs in a game are positive reviews, suggesting that having bugs in a game does not necessarily lead to negative reviews. We examined the negative reviews with bugs, and observed that most of the reported bugs in negative reviews can block players from playing or finishing the game, i.e., they are severe bugs. The most common reported bugs are:

1. Incompatibility (e.g., “Works very very badly on windows 8... Unplayable.”)
2. Crashes (e.g., “works well until a large battle then it crashes. Want to play it but this makes it impossible”, “game keeps crashing”)
3. Bugs blocking users from playing (e.g. “Well it’s been months of trying to get this game to work, but it still doesn’t.”, “...you get back to the map and it just freezes and won’t accept user input...”)

We observed the following most common types of bugs in positive reviews:

1. Performance issues (e.g., “Laggy....”)
2. Audio or visual issues (e.g., “Audio tends to fade in and out sporadically and frames drop at specific sequences.”)

3. Crashes (often accompanied by a compliment about game design, e.g., "It is a very interesting game ... The game crashes sometimes.")

Our earlier analysis (see Chapter 4) observed that 64% of the urgent updates of games address feature malfunctions of games (e.g., "Fixed save game does not save your minibike"), most of which are not bugs that block users from playing or finishing the game. As urgent updates cause unnecessary stress on the development team, this finding suggests that developers can re-consider the priority of non-gameplay-blocking bugs, and reduce the number of urgent updates for non-gameplay-blocking bugs by delaying them and bundling them with regular updates.

Negative reviews contain more valuable information about the negative aspects of a game for developers. Table 6.9 shows that negative reviews have a higher portion of both "Con" and "Bug", and a lower portion of "Not helpful" reviews, indicating that negative reviews may provide developers with more valuable information about the negative aspects of the current game design. The finding agrees with reported results for mobile app reviews (Hoon et al., 2012), that low-star ratings provide more valuable information to developers.

Positive reviews also provide useful information. Table 6.9 shows that 29% of the positive reviews discuss cons of the games, and 7% of the positive reviews report bugs in the games. Moreover, positive reviews contain a higher portion of pros of the games, and a slightly higher portion of suggestions, than negative reviews. Knowing what players appreciate about a game is important for developers, as they can ensure that these pros remain or are further improved in future updates. For example, knowing what users consider the pros of a game can help developers to decide whether a feature can be removed. Hence, developers and researchers should not dismiss the information that can be extracted from positive reviews.

Early access games receive more bug reports and suggestions. Table 6.9 shows that early access reviews have a higher percentage of bug reports, and almost five times the percentage of suggestions of non-early access reviews. These numbers are reasonable considering that the purpose for developers of using the early access model is to gather more early feedback. The finding complements our earlier analysis, in which we show that games have a much more active discussion forum in their early access stage (see Chapter 5).

Indie games receive more suggestions than non-indie games. Table 6.9 shows that indie games receive a higher percentage of suggestions in reviews, as well as a higher percentage for both pros and cons of the games. A possible explanation is that the player community of indie games is more engaged than the community of non-indie games.

Summary: We identified 6 categories of reviews. Although negative reviews contain more valuable information for developers, the portion of useful information in positive reviews should not be ignored by developers and researchers. Players appear to value game design over software quality (i.e., the number of bugs in a game).

6.4.2 RQ2: How long do players play a game before posting a review?

Motivation: We studied the number of playing hours before posting a review, as this number is a unique attribute that is not found in mobile app reviews. Prior work has shown that the first sustained play session is important for players' engagement (Cheung et al., 2014). With the number of playing hours associated with each review, researchers can quantify and study the importance in depth, and provide developers with suggestions for designing the storylines and levels of a game accordingly. In addition, some online distribution platforms (e.g., Nintendo Game Store[7]) have a minimum requirement for the playing time before allowing a user to post a review (Machkovech, 2018), suggesting that reviews with the same rating but different usage times may have different values. The findings of this section can demonstrate the value of studying app usage time to mobile app developers and researchers.

Approach: We use the Wilcoxon rank sum test to compare the distributions of playing hours across different types of reviews and reviews from different types of games, as explained in Section 6.2.4. The Wilcoxon rank sum test is the unpaired version of the Wilcoxon signed-rank test that we used in Section 6.3.1. As the reviews with playing hours that are used in this RQ were crawled across all the studied games, we use an unpaired test to compare the distributions. We use Cliff's delta effect size to quantify the difference in the distributions. Table 6.10 shows the description of the dataset that is used in this RQ.

Findings: **Gamers play a game for a median of 13.5 hours before posting a review.** Figure 6.12 shows the distribution of the playing hours that are associated with each review. The distribution has a median of 13.5 hours, indicating that half of the reviews are posted within the first 13.5 hours of playing.

Negative reviews are posted after significantly less playing hours than positive reviews. Figure 6.13 shows the distributions of playing hours for positive reviews and negative reviews. There are 21,874 positive reviews and 6,285 negative reviews in the

[7]`https://www.nintendo.com/games/`

Table 6.10: Dataset description of our study of playing hours associated with each review

	Indie	**Non -indie**	**Early access**	**Non-early access**	**Free -to-play**	**Non-free -to-play**
# of games	4,721	3,304	786	7,239	386	7,639
# of positive reviews	9,329	12,624	3,685	18,268	3,567	18,386
# of negative reviews	2,419	3,919	1,333	5,005	1,016	5,322
# of all reviews	11,748	16,543	5,018	23,273	4,583	23,708

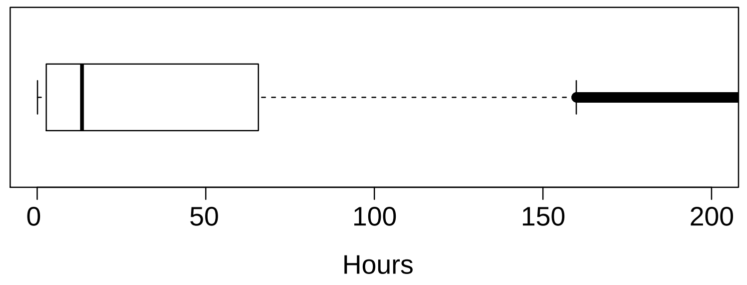

Figure 6.12: The distribution of playing hours that are associated with each review

dataset that is used in this RQ. The playing hours for negative reviews are significantly less than the positive reviews, with a small effect size. The median number of playing hours for positive reviews is 15.5 hours, while the median number of playing hours for negative reviews is 6.6 hours. Hence, we suggest that developers should be extremely cautious about the design of the gameplay for the first 6.6 hours, as more than half of the negative reviews are made within the first 6.6 hours of playing.

We also observe a higher peak in the distribution of playing hours for positive reviews, and a more flat distribution of playing hours for negative reviews. One possible explanation is that there may be different reasons that lead to negative impressions which occur at different period of gameplay of a game. To understand more about why people complain about a game even though they played it for a long time, we manually

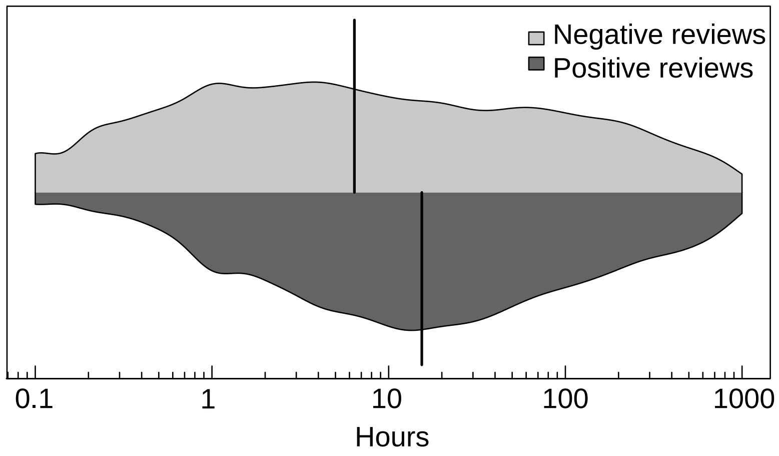

Figure 6.13: The distributions of playing hours for positive and negative reviews. The vertical lines represent the median. The distributions are significantly different ($p < 0.05$), with a small effect size.

examined the 63 negative reviews with the longest 1% playing hours. We observed that many of the players who posted such reviews are actually satisfied with the general idea of the game. However, the gaming community (e.g., players who ruin the gameplay), the quality of the latest updates, or the pricing of Downloadable Contents (DLC) of the games disappointed these loyal players, indicating that a badly maintained gaming community, or a poorly planned update may ruin the loyalty of the player base.

We also manually examined the 210 negative reviews with equal to or less than 0.1 playing hour, as 0.1 hour is the smallest granularity at which we monitor playing hours. We identified two major reasons for users to give a negative review after such a short playing time:

1. Severe bugs (e.g., “Doesn’t even log into the game”, “Game crashes every time I try to start it”).

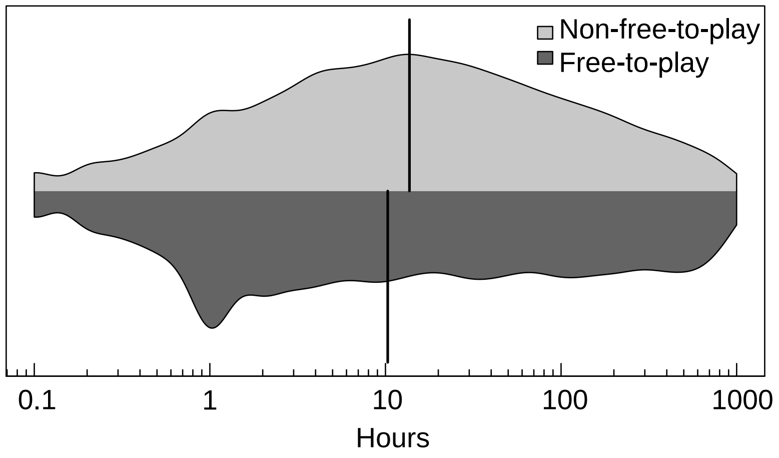

Figure 6.14: The distributions of playing hours that are associated with reviews for free-to-play and non-free-to-play games. The vertical lines represent the median. The distributions are significantly different ($p < 0.05$), with a negligible effect size.

2. Bad design of the game. (e.g., “Gameplay is so boring”, “Not particularly engaging”).

A peak in the number of reviews of free-to-play games is observed after approximately one hour of playing. Figure 6.14 shows the distributions of playing hours for reviews of free-to-play games and non-free-to-play games. The Wilcoxon rank sum test shows a significant difference between the two distributions, with a negligible effect size. It is worth noting that Figure 6.14 shows a density peak at around one hour for free-to-play games, indicating that many free-to-play game players make their judgement of a game after one hour of playing. Possible explanations are that (1) free-to-play games are shorter, or (2) players give up sooner as they did not invest money to buy the game. Moreover, we also observed a different intensity of the peaks around one hour across other dimensions in Figure 6.13 and Figure 6.15, suggesting that the first hour of game design is very important.

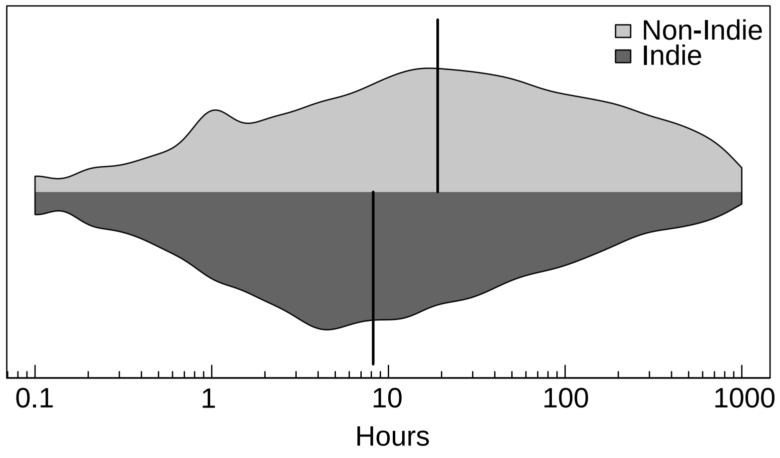

Figure 6.15: The distributions of playing hours that are associated with reviews for indie and non-indie games. The vertical lines represent the median. The distributions are significantly different ($p < 0.05$), with a small effect size.

We calculated that the median length of the reviews with 1±0.5 playing hours is 39 characters, while the median length of the reviews with 10.3±0.5 playing hours (the median playing hours of free-to-play game reviews) is 38 characters. For non-free-to-play games, the length is 64 and 69 respectively, indicating that reviews that are posted around the first playing hour do not necessary contain less information. The median lengths are shorter than our finding in Section 6.3.1 because we did not group the reviews by games in this RQ. The median review length (without grouping the reviews per game) for all the reviews studied in Section 6.3.1 is 80.

Our finding on the importance of the first playing hour agrees with the work by Cheung et al. (2014), which hypothesized that the “first hour experience” is critical for players’ engagement. While Cheung et al.’s work uses “first hour experience” to refer to the first sustained play session, our finding confirms from actual empirical data that the first few hours are important for user experience but also suggests that the first hour is even more important for free-to-play games.

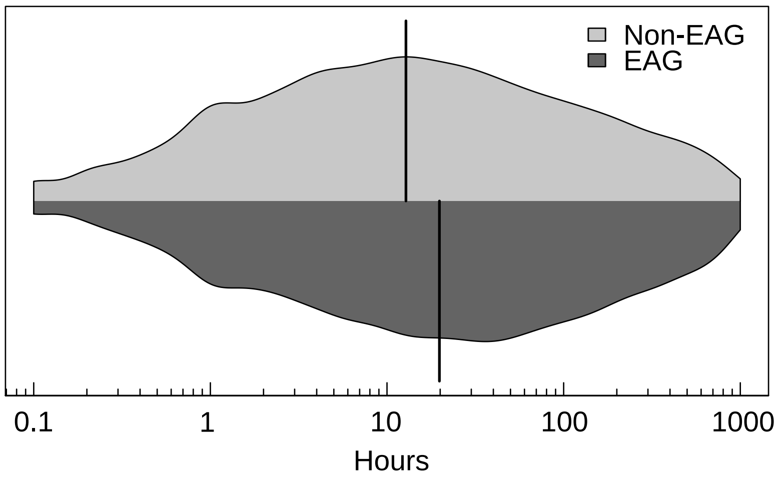

Figure 6.16: The distributions of playing hours that are associated with reviews for EAG and non-EAG. The vertical lines represent the median. The distributions are significantly different ($p < 0.05$), with a negligible effect size.

Indie game developers have a shorter time to satisfy players in their games than non-indie game developers. Figure 6.15 shows the distributions of playing hours for reviews of both indie games and non-indie games. The Wilcoxon rank sum test shows that the playing hours that are associated with reviews for indie games are significantly shorter than non-indie games, with a small effect size. Hence, indie game developers have less time to satisfy players than non-indie game developers. A possible explanation is that indie games may have a shorter storyline. It is worth noting that among the 3,628 studied indie games, only 183 (5%) of them are free to play.

Players of games in the early access stage spend more time playing a game before posting a review. Figure 6.16 shows the distributions of playing hours for early access reviews and non-early access reviews. The Wilcoxon rank sum test shows that the playing hours that are associated with early access reviews are significantly longer than non-early access reviews. However, the effect size is negligible. This finding agrees

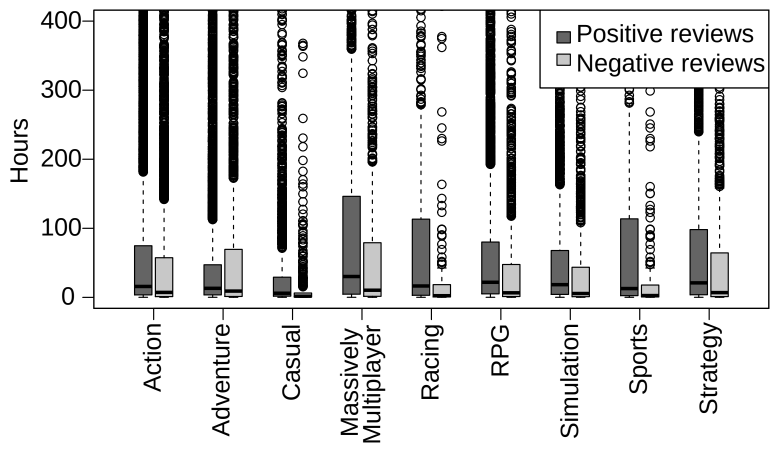

Figure 6.17: The distributions of playing hours that are associated with reviews for games in each genre. Outliers greater than 400 are removed for better demonstration.

with our earlier analysis (see Chapter 5), which suggests that players of early access games tend to be more tolerant of the quality of a game during its early access stage.

Players of casual games spend the least time playing a game before posting a review. Figure 6.17 shows the distributions of playing hours for each game genre. Casual games have the lowest range of playing hours, indicating that casual game players make their judgement about a game faster than other genres. The genre with the highest median playing hours is the Massively Multiplayer game. In addition, the median number of playing hours for negative reviews is lower than for positive reviews for all game genres.

Summary: Gamers play a game for significantly less time before posting a negative review than before posting a positive review. The first hour playing experience is more important for free-to-play games. Developers should pay particular attention to the design of the first 6.6 hours of gameplay, as the majority of negative reviews are posted within that period.

6.4.3 Implications of our Findings

In this section, we discuss the implications of our findings for researchers and future studies that focus on reviews of online distribution platforms (e.g., mobile app stores and the Steam platform).

Reviews for games are different from reviews for mobile apps. Throughout our study of reviews for games on the Steam platform, we observed that in several aspects, game reviews are different from mobile app reviews. Firstly, the median number of reviews received by games per day (2) is much lower than the number for mobile apps (22). However, game reviews are longer than mobile reviews. Secondly, prior research on mobile app reviews only had access to the number of owners of an app, while our study analyzed both the number of owners and the number of active users (players) of games, and observed that the number of players has a stronger relation with the number of reviews received per day. In addition, prior mobile app studies that do study paid apps, tend to ignore the impact of discounts. These studies of mobile app reviews should be revisited, as we observed that a sales event has the strongest relation with an increase in the number of received game reviews. Finally, games receive a higher percentage of reviews that contain useful information for a developer than mobile apps. Future studies should investigate further whether existing methods for analyzing mobile app reviews can be applied to game reviews. Two observations that we made during our manual analysis of game reviews, which limit the applicability of the automated analysis tools that are currently popular in mobile app review research, are that (1) game reviews tend to contain sarcastic language and that (2) game reviews tend to contain game-specific terminology. As a result, many automated techniques for analyzing natural language do not achieve a high accuracy on game reviews.

Different types of games have different reviews. In our study, we compared all the studied reviews along four dimensions: positive and negative; indie and non-indie; early access and non-early access; free-to-play and non-free-to-play. We observed that the median length of reviews is significantly different along every studied dimension. We also noticed that reviews provide different types of information to developers along every studied dimension. For example, indie games receive more suggestions than non-indie games. Therefore, future studies of game reviews should consider the impact of different types of games.

Information that can be extracted from positive reviews should not be ignored by future studies. Previous studies in mobile app reviews often focus on the negative side of the reviews (Khalid et al., 2015; McIlroy et al., 2016b), and rarely consider positive reviews, or the praise in reviews (Panichella et al., 2015). However, we observed that 29% of the positive reviews discuss cons of the games, and 7% of the positive reviews report bugs in the games despite their positivity. Positive reviews also contain a higher portion of suggestions than negative reviews. In addition, knowing about what players appreciate in a game can help with making important decisions about the evolution of a game. Hence, the helpful information in positive reviews should not be ignored by future studies.

The number of playing hours before posting a review provides a unique and helpful insight for developers. In Section 6.4.2, we showed that the number of playing hours associated with game reviews is a unique attribute that is not found in mobile app reviews. This attribute yields new information that can be leveraged by game developers for game design. We know from prior work (Cheung et al., 2014) that the design of the initial gameplay is important. Using the data from the Steam platform, we can quantify and study this importance in depth for all types of games. In this chapter, we showed that the number of playing hours provides useful information. Future longitudinal studies should be done to draw definitive conclusions about how gameplay is correlated with the playing time.

In addition, our study sheds light on the fact that reviews that have the same rating, but are posted after different usage times, may provide different information. For example, negative reviews on the Steam platform that are posted after many playing hours, are usually associated with a bad community or update, while negative reviews with few playing hours are usually caused by severe bugs or bad design. Unfortunately, current mobile app stores do not provide the usage time with reviews. As a result, prior studies on mobile app reviews treat all reviews with the same rating equally. However, our findings show that it could be beneficial for researchers and developers to collect and analyze the usage time for mobile apps as well. Hence, mobile app stores and developers should consider integrating the usage time into mobile app reviews. For example, mobile app stores could identify "early" and "late" reviews, to give developers and other mobile app users more context about a user's opinion.

6.5 Threats to Validity

This section presents the threats to the validity of our findings.

6.5.1 Internal Validity

A threat to the validity of our findings is that we only studied reviews that were written in English for the research questions that involve the contents of reviews. However, there is an obvious limitation in reading reviews in all languages. Future studies should validate whether our observations hold for non-English reviews.

Although on the platform level, there are no incentives for gamers to write reviews, there may exist games that provide an in-game incentive (making it hard for us to find out without actually playing the game). The incentives could possibly bias our results.

To understand what are gamers talking about in reviews, we manually categorized a statistically representative random sample of English reviews. Our sample size is 472 reviews for English reviews, which has a confidence level of 95% and a confidence interval of 5; and 96 reviews for each categories of reviews that we studied, which has a confidence level of 95% and a confidence interval of 10. Although the sample size is relatively small compared to the population of 6,768,768 English reviews, our sample is statistically representative of the whole population of game reviews on Steam.

Another threat to the validity of our findings in Section 6.4.2 is that we only collected one month of reviews that have an accurate number of playing hours. Future studies are needed on a larger dataset to verify our findings. In addition, as there may be other factors influencing player's playing hours, such as the player's expectation based on the hype that was created for a game, our findings do not suggest causations.

We conducted manual analysis to understand the content of reviews in Section 6.4.1. We have attempted to apply latent Dirichlet allocation (LDA) (Blei et al., 2003) to automatically extract topics from reviews. However, we did not get meaningful topics. A possible explanation is that players use a large amount of game-specific terminology in their reviews, which limits the applicability of LDA. In addition, although the two researchers had a partially different categorization for 85 out of 472 reviews, the vast majority of the disagreements were cases in which one of the researchers assigned an additional label to a review (hence they were no major conflicts). The conflicts were resolved by discussing and coming to an agreement.

Other threats to the validity of our study concern the metrics that were used in Section 6.3.2 in our model for the game-specific characteristics that are related to the number of reviews that are received each day. The number of owners used in our study were estimated from a representative sample by Steam Spy. Although a three-day rolling sample was used to increase the accuracy, there can still exist a deviation from the actual number of owners. However, because the sales data is confidential in the gaming industry, this is the most accurate method to our knowledge to estimate the number of owners of a game. In addition, we estimated the lifetime of games using the release date as advertised on the Steam Store page. This number is an estimation because developers are allowed to change that release date. We observed that for some games that already existed before they were released on Steam, developers changed the release date to the real release date. We do not have data (reviews, price, etc.) between the real release date and the date that the game was released on Steam. However, we expect it is sufficiently accurate to be used to give a good estimation of the lifetime of games.

6.5.2 External Validity

In our empirical study, we studied the reviews on Steam. The findings of our study may not generalize to other reviews on different distribution platforms. However, as stated in Chapter 2, Steam is the largest digital distribution platform for PC gaming. Hence, the reviews on Steam are representative for a large number of reviews. We also compared our findings on game reviews to mobile app reviews where possible.

6.6 Chapter Summary

The competition within the gaming industry, and the hard-to-please user base has made the quality of games an increasingly important issue. As game reviews are a direct reflection of user concerns, a better understanding of reviews can help developers produce games with an improved user-perceived quality.

In this chapter, we performed an empirical study on the reviews of 6,224 games on the Steam platform. We studied the number and the complexity of reviews, the type of information that is provided in the reviews, and the number of playing hours before posting a review.

The most important findings of our study are:

1. Negative reviews are often posted after only half of the playing hours of positive reviews.
2. A large number of reviews for free-to-play games are posted after approximately one hour of playing.
3. Players complain more about game design than bugs in their reviews.
4. Although negative reviews contain more valuable information for developers, the useful information in positive reviews should not be dismissed.
5. Game reviews are different from mobile app reviews along several aspects.

Based on our findings, we provide the following suggestions for future studies:

1. Due to the difference we discovered between game reviews and mobile app reviews, future studies should investigate further how to adjust existing methods for analyzing mobile app reviews to apply them to game reviews.

2. When studying game reviews, the impact of different types of games should be considered.

3. Information that can be extracted from positive reviews should not be ignored by future studies.

4. The number of playing hours before posting a review should be included in future studies, as the attribute provides a unique insight into how a player's opinion is related to their playing time. Mobile app stores and developers should consider integrating the usage time into mobile app reviews.

We believe that our findings and suggestions can help researchers conduct future studies in game reviews, and in turn help developers improve the user-perceived quality of their games. Future studies should investigate advanced methods that help developers who are not able to read all reviews of their games each day, such as developers of the top 4% games, or developers who have an additional full-time job. Furthermore, our findings show that game reviews are different from mobile app reviews in many aspects, and reveal several important factors that are often ignored by mobile app researchers (e.g., the impact of discounts on the number of reviews). Prior work on mobile app reviews needs to be revisited while considering such factors into account.

In the following chapter, we study the gameplay videos on the Steam platform to demonstrates the value of gamer-produced audiovisual content for software engineers.

CHAPTER 7

Studying Gameplay Videos on the Steam Platform

In Chapter 6, we studied the game reviews on the Steam platform. In addition to game reviews, the Steam Community also allows gamers to share audiovisual content, such as gameplay videos. In this chapter, we conduct an in-depth study of gameplay videos that are posted by gamers. In particular, we study whether gameplay videos can be used as a supplemental source of bug reports for game developers. We first conducted a preliminary study to understand the number of gameplay videos on the Steam platform, and the difficulty of identifying bug reports from these gameplay videos. We show that naïve approaches such as using keywords to search for bug videos are time-consuming and imprecise. We propose an approach which uses a random forest classifier to rank gameplay videos based on their likelihood of being a bug video. Our proposed approach achieves a precision that is 43% higher than that of the naïve keyword searching approach. In addition, our approach has both a mean average precision at 10 and a mean average precision at 100 of 0.91. Our approach can also be generalized to gameplay videos that are available on other platforms. Our study demonstrates the value of gamer-produced audiovisual content for software engineers, and helps game developers leverage readily available gameplay videos to automatically collect otherwise hard-to-gather bug reports for games.

7.1 Introduction

DESPITE the large effort that is spent on Quality Assurance (QA) in the development process of a game, it is not possible to discover or fix all bugs in a game before releasing it to the market. Our earlier analysis (see Chapter 4) shows that 80% of the Steam games release urgent updates to fix issues such as feature malfunctions or game crashes. A common practice to ensure the user-perceived quality of a game is to allow gamers to submit bug reports that describe any encountered bugs during gameplay. Many studios and games have discussion forums or in-game features for gamers to report bugs (e.g., Blizzard (Sixen, 2010), EVE Online (Habakuk, 2017), League of Legends (Afic, 2015)). The discussion forums and in-game bug reporting features usually require gamers to provide a title for the bug, a detailed description of the bug, and the necessary steps to reproduce the bug. However, this practice is dependent on bug reporters, and therefore hard to be automated. In addition, the reported bugs can be hard to reproduce, often due to the lack of reproduction steps in the report. Zimmermann et al. note that valid and accurate steps to reproduce a bug are considered the most important part of a bug report by developers, yet such information is difficult to provide by the bug reporters (Zimmermann et al., 2010). To automate the process of bug information collection, and provide more context to the bug report, some software systems integrate the functionality of crash report generation. However, this functionality is intrusive and privacy-sensitive (Castro et al., 2008).

One possibility to learn more about the problems that gamers encounter is by studying gameplay videos (Lewis et al., 2010). In recent years, gameplay videos have become popular in the gaming community. Two of the top five YouTube channels with the highest number of worldwide subscribers are related to gaming (Petrova and Gross, 2017). The Steam platform also allows gamers to share and discuss their YouTube gameplay videos on the Steam Community website (Valve, 2018). Gamers share gameplay videos that cover several topics including game tutorials, game play-throughs (i.e., a video of playing a game from start to finish), and game bugs. These bug videos open up a new opportunity for developers to collect information about bugs. Several studios have realized the value of bug videos, and encourage gamers to submit relevant screenshots or videos along with their bug report (Sixen, 2010; Afic, 2015; Habakuk, 2017), especially for bugs in the game's graphics ("should probably always have a screenshot" (Sixen, 2010)) and general functionality bugs (which "may only be visible in movies" (Sixen, 2010)). However, the applicability of leveraging gameplay videos as a source for bug reports has not been investigated in prior studies.

In this chapter, we conduct an in-depth study of gameplay videos that are posted by gamers. In particular, we study whether gameplay videos can be used as a supplemental source of bug reports for game developers. We first conducted a preliminary study of the gameplay videos on the Steam platform, to understand the number of gameplay videos and the difficulty of distinguishing bug videos from other gameplay videos. We found that 21.4% of the games on the Steam platform receive more than 50 game videos, and up to a median of 13 hours of video runtime per day, and that a naïve keyword searching approach for identifying videos that showcase a game bug only achieves a precision of 56.25%. To help developers better leverage such bug videos, we propose an approach that uses a random forest classifier to further rank the videos with their likelihood of showcasing a game bug, using features that are extracted from the metadata of videos. Our evaluation shows that our proposed approach achieves a precision of 0.80, which is 43% higher than that of the naïve keyword searching approach. In addition, our proposed approach has both a mean average precision at 10 and a mean average precision at 100 of 0.91. Our study makes the following contributions:

- A demonstration of the value of mining video data that is produced by the gaming community.
- The first approach that showcases how to automatically identify bug videos with high precision.
- An analysis of the characteristics of videos that showcase a game bug.
- A labelled dataset of 1,400 bug videos[1] to encourage further studies in this important research direction.

7.2 Background on the Study of Gameplay Videos

In this section, we briefly describe YouTube and YouTube Gaming, and the videos on the Steam platform.

7.2.1 YouTube and YouTube Gaming

YouTube is one of Google's subsidiaries, and serves as the largest video-sharing website and the second-most popular website in the world (Alexa, 2018). YouTube allows users to view, upload, comment, rate, and share videos on various topics, such as video blogging, music videos, educational videos, and gameplay videos. When uploading a

[1] https://github.com/SAILResearch/replication-bug_videos

video to YouTube, the user will provide the title, description, keywords, and category of the video. YouTube provides a pre-defined set of categories for users to choose from, such as "Sports", "Education", and "Gaming". If a user selects "Gaming" as the category of the video, YouTube would ask the user to enter the title of the game that is associated with the video.

Videos on YouTube are aggregated into channels. Each user has their own YouTube channel, which contains the videos that they uploaded. In addition, YouTube automatically aggregates videos into system-generated channels, such as YouTube Gaming channels (YouTube, 2018). YouTube Gaming[2] is a site under YouTube that is dedicated to gaming videos and channels. When a user uploads a video under the "Gaming" category, and provides the title of the associated game, YouTube automatically adds the video into the YouTube-generated gaming channel for that game, and labels the viewing page of the video with the title of the game, with a link to the YouTube Gaming channel of the game. However, unlike user-owned channels, the system-generated channels for specific games do not provide an exhaustive list of all the videos under the channels.

7.2.2 Videos on the Steam Platform

The Steam Community allows gamers to share audiovisual content, such as screenshots and videos, under each game's community page. It is worth noting that the Steam platform does not provide a video storage service. To post a video for a game on the Steam Community, a gamer needs to upload the video to YouTube first, then authorize the Steam platform to access the gamer's YouTube account for the list of potential

[2]https://gaming.youtube.com/

Table 7.1: Dataset description of our study of gameplay videos

# of studied Steam games	16,252
# of unique video posts in the Steam Community	1,785,366
# of Steam videos that are available on YouTube	1,636,689
# of non-Steam games for verification	4
# of YouTube videos for non-Steam games	20,754

videos for posting. The gamer can then select videos that they wish to share, and associate the videos with games. The videos are then posted on the gamer's profile page on Steam. Gamers are allowed to edit the title and the description of a video post, which are filled by default with the video's YouTube title and description.

7.3 Data Collection of Our Study of Gameplay Videos

In this section, we describe how we extracted and processed data. In general, we collected metadata of videos for games on the Steam platform (i.e., Steam games) and for games that are not on the Steam platform (i.e., non-Steam games). Table 7.1 presents the description of our collected dataset. Figure 7.1 gives an overview of our process.

7.3.1 Collecting videos for Steam games

To collect videos for Steam games, we first collected a list of all Steam games from the Steam Store. We took a snapshot of all the 16,252 games that were available in the Steam Store on August 1st, 2017, using a customized crawler.

We then developed a customized crawler that collects video posts for each Steam game in the Steam Community. For each Steam game in the collected game list, the crawler visited its Steam Community page, and collected all the video posts on the

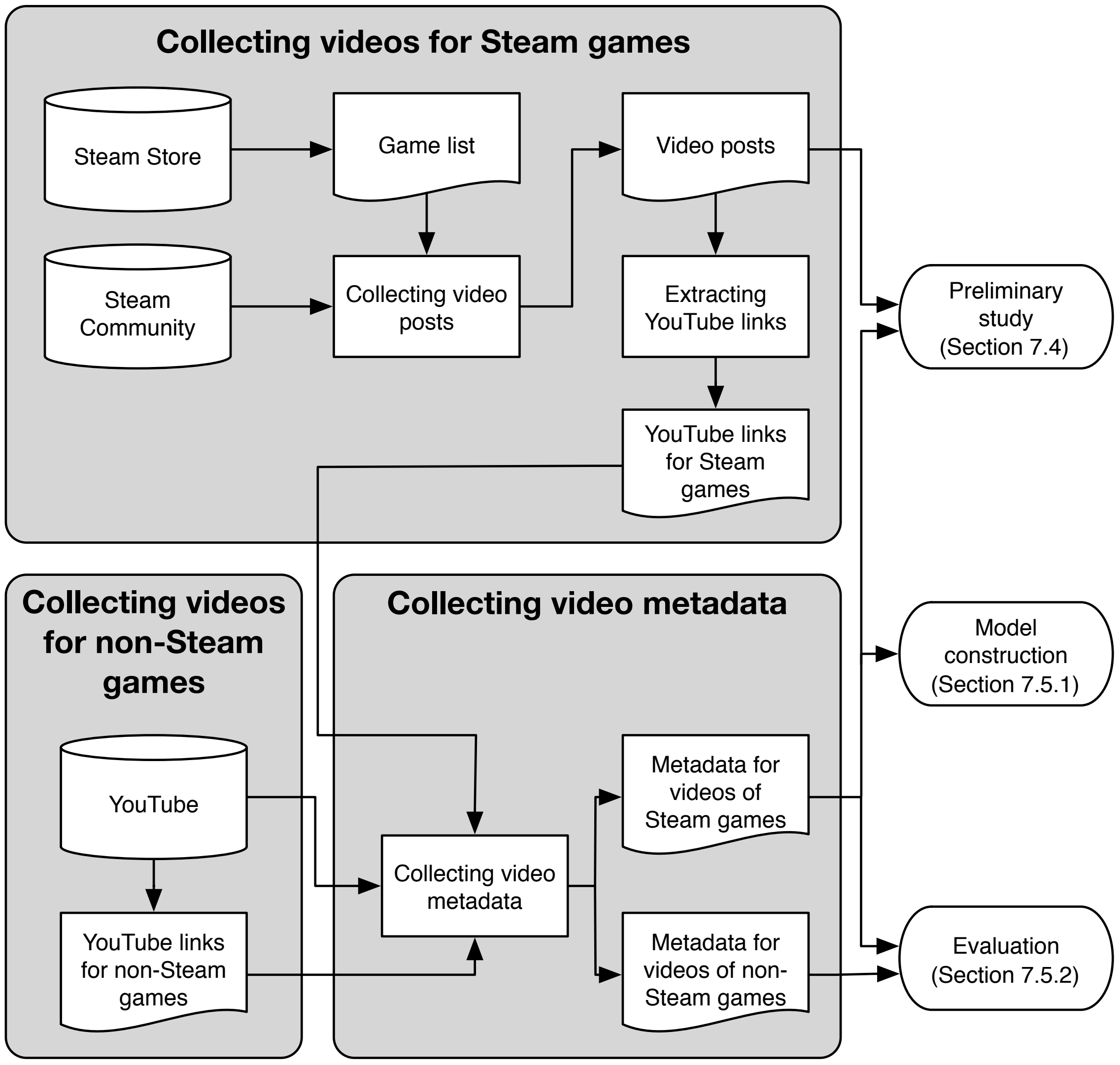
Collecting videos for Steam games
Steam Store
Game list
Video posts
Steam Community
Collecting video posts
Extracting YouTube links
YouTube links for Steam games
Preliminary study (Section 7.4)
Collecting videos for non-Steam games
Collecting video metadata
YouTube
YouTube links for non-Steam games
Collecting video metadata
Metadata for videos of Steam games
Metadata for videos of non-Steam games
Model construction (Section 7.5.1)
Evaluation (Section 7.5.2)

Figure 7.1: Overview of data collection process for the study of gameplay videos

Steam Community for that game. Specifically, the crawler collected the following information for each video post: title, description, and the YouTube link of the video. The crawler also recorded the specific Steam game to which a video post belongs. Table 7.2 gives an example of a video post.

Table 7.2: An example of a video post

Title	Fallout New Vegas Hardest punch
Description	I hit him so hard he broke
YouTube link	`https://youtu.be/-TiTf5G4wpg`
Associated Steam game	Fallout: New Vegas[1]

[1] `https://store.steampowered.com/app/22380/`

In total, we collected 1,989,140 video posts on the Steam Community for all the games. We noticed that there were some video posts with the same YouTube links (i.e., different video posts for the same video). We observed that the video posts with duplicated YouTube links are posted by the same Steam accounts under the same game, potentially by mistake. We removed such duplicate video posts. We collected a total of 1,785,366 unique YouTube links for videos of Steam games.

7.3.2 Collecting videos for non-Steam games

To evaluate the generalizability of our results, we used a customized crawler to collect the YouTube videos for games that were not released on the Steam platform. We focused on games that were published by Electronic Arts (EA), as many recent AAA games[3] from EA were exclusively released on EA's own online distribution platform for PC games, the Origin platform (Electronic Arts Inc., 2018). We selected 4 EA Origin exclusive games that have a large number of gameplay videos on YouTube, i.e., FIFA 16, FIFA 17, NHL 16, and NHL 17. As the YouTube Gaming channels do not provide a comprehensive list of all videos under a game's channel, the crawler first used the search function provided by YouTube to search for videos with keywords being the title of the game, then it visited the page of each video in the search result to confirm whether the

[3] Games with the highest budgets for development and promotion.

video is associated with the target game. In total, we collected 20,754 videos for the 4 studied games that are not on the Steam platform.

7.3.3 Collecting video metadata

For each YouTube video collected in Section 7.3.1 and Section 7.3.2, we used a customized crawler to collect the video's metadata on YouTube. In particular, we collected the title, category, description, tags, and length (in seconds) for each video.

We noticed that 148,677 YouTube links from the Steam Community were not accessible on YouTube at the time of crawling, due to the following reasons:

- The uploader deleted the video
- The video violated copyright from a third party company
- The video is being processed (e.g., transcoded)
- The video is private
- The video violated YouTube's Terms of Service
- The video is not available in the country from which our crawler runs (i.e., Canada)
- The YouTube account that is associated with the video has been terminated

In total, we collected the metadata for 1,657,443 YouTube videos (1,636,689 for Steam games and 20,754 for non-Steam games), which were uploaded by 323,325 gamers. For the videos of the Steam games, we linked each video's metadata from YouTube to the specific video post on the Steam Community.

7.4 Preliminary Study of Gameplay Videos on the Steam Platform

As there exists no prior research on mining gameplay videos from the Steam platform, in this preliminary study, we aim at providing an overview of such videos. In addition, as shown in prior work (Lewis et al., 2010), gameplay videos contain valuable information for game developers, such as game bugs. In our preliminary study, we examined the effort that is needed for developers to identify gameplay videos that showcase a game bug (i.e., *bug videos*) on the Steam platform.

Approach: We examined the effort that is required from developers to identify bug videos using two naïve approaches:

1. **Watch all videos of a game.** To identify bug videos from the pool of all posted videos of a game on the Steam platform, the most accurate way is to manually watch all the videos of a game. As the bug may show up at the end of the video, it is necessary to watch the whole video to locate the bug. Hence, we calculated the number of videos and the accumulated video length for each game, to understand the effort that would be needed by developers to watch all the videos of their games.

2. **Search for keywords.** Another naïve approach to identify bug videos is to search for the occurrence of certain keywords in the video metadata (e.g., in the title, description, and tags). Hence, we applied a naïve keyword searching approach to game videos to evaluate its performance, and to understand if such an approach would be effective for developers.

We consider the problem of identifying bug videos as a binary classification problem, i.e., we classify all the videos of a game into two classes: bug videos (True), or not bug videos (False). Hence, we have the following confusion matrix:

		Predicted	
		Bug videos	Not bug videos
Actual	Bug videos	True Positive (TP)	False Negative (FN)
	Not bug videos	False Positive (FP)	True Negative (TN)

The precision of an approach to identify bug videos can be calculated as $\frac{TP}{TP+FP}$. We use precision instead of recall to evaluate the performance of an approach because we consider bug videos as a supplemental source of bug reports (in addition to textual issue reports). Therefore, we prefer missing actual bug videos over wrongly identifying videos as bug videos, so that developers would not waste time watching non-bug videos.

There are five text fields in the metadata we collected for videos that are used for the keyword search: the Steam video post title and description, and the YouTube title, description, and tags. Note that the Steam and YouTube title and description are not always the same.

To find suitable keywords to search for bug videos, we used the following process:

1. **Removing stop words.** Stop words (e.g., “the”) have a high occurrence in the studied text and are irrelevant to the study. We removed the stop words from the text fields in the metadata using the stop word list provided by the Natural Language Toolkit (NLTK) (NLTK Project, 2017).

2. **Stemming.** To avoid different forms of the same word (e.g., “argue”, “argued”, “argues”, “arguing”) being recognized as different words, we stemmed (e.g., “argu”)

Table 7.3: Selected keywords for matching

Groups	Keywords (# of videos)	Description
Bug	bug (32,166), glitch (28,025)	Directly related to bugs
Hack	hack (28,459), hacker (8,405), cheat (21,639), cheater (6,336)	Hackers / cheaters in a game make use of game bugs

all the words in the text fields in the metadata using NLTK. As the stop word list from NLTK is not stemmed, we apply stemming after removing the stop words.

3. **Removing uncommon words.** Words occurring in a very small number of videos may be game-specific and hence not helpful for identifying bug videos across all games. For each remaining word, we counted the number of videos that contain the word in their metadata. Then we kept the words that occur in more than 1,636 videos (0.1% of the number of collected videos that are available on YouTube).

4. **Selecting bug-related keywords.** We manually checked each of the words and selected a list of keywords that are related to bugs in games. Table 7.3 shows the keywords that we selected (lower case, singular form).

When selecting the keywords, we noticed an interesting keyword “wtf”, which occurs in 16,649 videos’ metadata. “Wtf” is a unique genre of gameplay videos that record surprising moments in the gameplay, either an unexpected event or an impressive operation. We did not include “wtf” in our keyword list, as it is not directly relevant to game bugs. However, it may be interesting for developers to include this keyword, if their resources for watching these videos permit to do so.

Findings: **21.4% of the studied games have more than 50 videos on the Steam platform.** Figure 7.2 shows the distribution of the number of videos for each game. Games on the Steam platform have a median of 9 videos. 3,483 (21.4%) games have

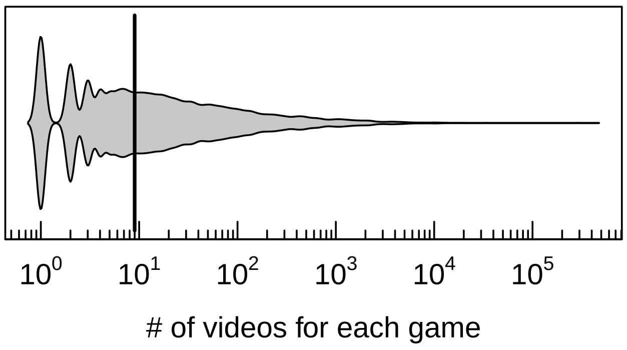

Figure 7.2: The distribution of the number of videos of each game. The vertical line shows the median value of the distribution.

each received more than 50 game videos. The game with the most game videos on the Steam platform is the *Counter-Strike: Global Offensive* game, for which 352,443 game videos were posted.

26.2% of the games on the Steam platform received an accumulated length of more than 10 hours of game video, with a longest accumulated length of over 3 years. Figure 7.3 shows the distribution of the accumulated length of game videos for each game. Games on the Steam platform received a median accumulated length of 136 minutes (2.3 hours) of game videos. We calculated that 3,026 (26.2%) games receive an accumulated length of more than 10 hours, and 1,926 (16.7%) games receive an accumulated length of more than 24 hours (1 day) of game videos. The longest accumulated length of game videos received by one game is 1,593,005 minutes (3.03 years), also for the *Counter-Strike: Global Offensive* game.

To further demonstrate the workload for developers to watch gameplay videos everyday, we calculated the accumulated length of videos for each game per day. Figure 7.4 shows the distribution of the median accumulated length of videos per day. The

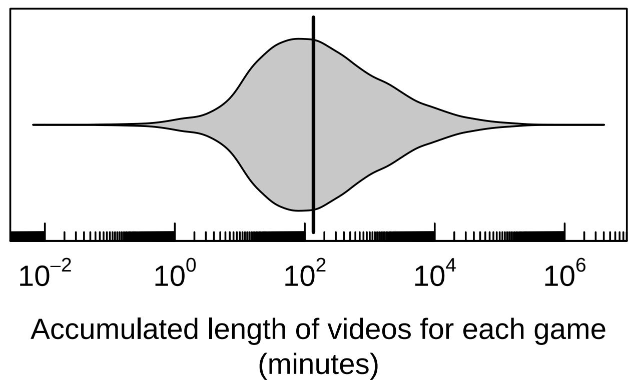

Figure 7.3: The distribution of the accumulated length of videos of each game. The vertical line shows the median value of the distribution.

median value of the distribution is 16.38 minutes, with a maximum of 13.27 hours, suggesting that in general developers of a game will receive around 16 minutes of videos in a day; and in extreme cases, developers can receive a median of up to 13 hours of gameplay videos per day.

Using keywords to identify bug videos has a precision of 56.25%. We applied the keyword searching approach to all 1,636,689 gameplay videos for Steam games that are available on YouTube. In total, the keywords matched 83,321 videos. We randomly selected a statistically representative sample of 96 videos from the videos that matched the keywords (with 95% confidence level and 10% confidence interval), and manually verified whether they are bug videos. In total, 54 out of these 96 videos are bug videos. Hence, the precision of a naïve keyword searching approach for identifying bug videos is 56.25%. We manually checked the false positives to understand the reason behind the low precision of the keyword searching approach, and we extracted the following cases for the false positives:

- Advertising other bug videos under the same channel in the description

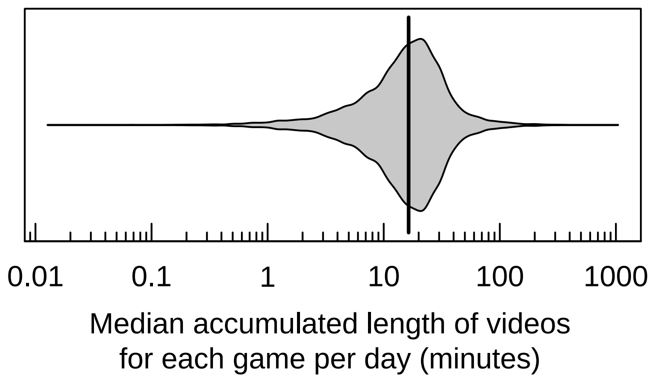

Figure 7.4: The distribution of the accumulated length of videos of each game per day. The vertical line shows the median value of the distribution.

- Stuffing irrelevant keywords (Google, 2018) in the description that include our keywords
- Videos about games that themselves involve the keywords (e.g. *Age of Barbarian*: a hack and slash game[4]; *Bug Butcher*: a game about catching bugs[5])

The majority of videos of most games are posted by a few gamers, while each gamer only posts videos for a few games. For each game, we sum up the number of video posts that are posted by the top 3 gamers with the highest number of video posts, and calculated the ratio of the sum to the total number of video posts the game received. Figure 7.5 shows the distribution of the ratio.

As shown in Figure 7.5, games receive a median of 75% of videos from their top 3 gamers. We further calculated the number of games that each gamer posted videos for. We found that 81% of the gamers only posts videos of one game, and 96% of the gamers only posts videos of no more than 5 games. To compare, gamers own an average 85.92

[4]https://www.youtube.com/watch?v=W9in3AEYuJk
[5]https://www.youtube.com/watch?v=pnFuzq8Ii8o

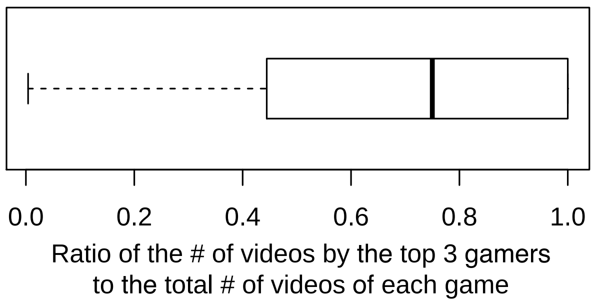

Figure 7.5: The distribution of the ratio of the number of videos by the top 3 gamers to the total number of videos of each game.

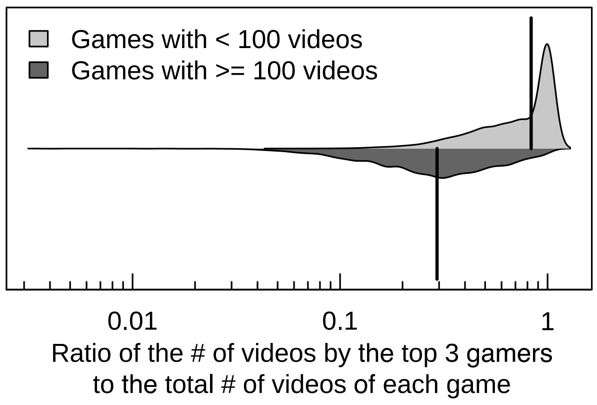

Figure 7.6: The distribution of the ratio of the number of videos by the top 3 gamers to the total number of videos of games with less than 100 videos and at least 100 videos.

games on the Steam platform (Galyonkin, 2018). Hence, they appear to post videos only for a small fraction of their games.

We compared the ratio between the games with at least 100 videos (which is 14% of the studied games), and the games with less than 100 videos. Figure 7.6 shows that the top 3 gamers contributed a median of 83% of the videos of a game with less than 100 videos, and a median of 29% of the videos of a game with at least 100 videos. The Wilcoxon rank sum test (see Section 4.4) confirms the significant difference with a large Cliff's Delta effect size.

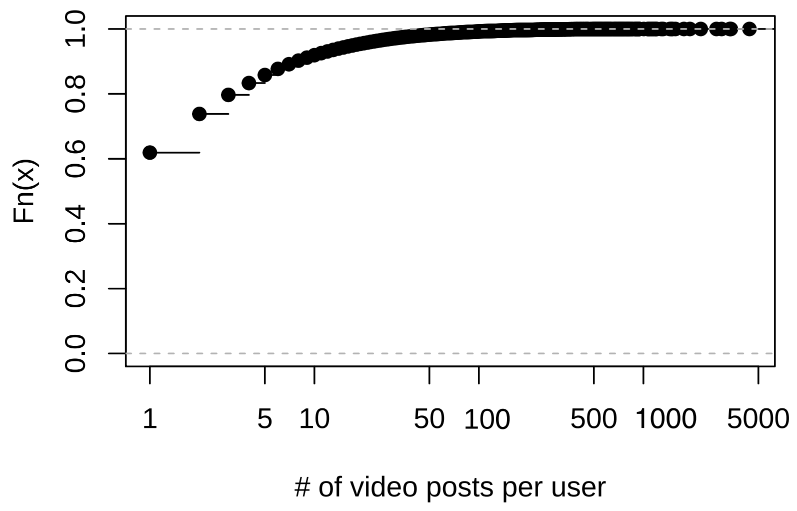

Figure 7.7: The empirical cumulative distribution function (Fn(x)) of the number of video posts per gamer.

The *GameGlitches* account contributes the most videos to the Steam Community. Figure 7.7 shows the empirical cumulative distribution function of the number of videos contributed by each account. 62% of the gamers posted only one video to the Steam Community. We calculated that 92% of the gamers posted at most 10 videos. The account with the highest number of videos is *GameGlitches*[6], with 4,413 videos. The majority of the videos under the *GameGlitches* account focuses on the glitches of games.

[6]http://steamcommunity.com/id/GamesGlitches/

Summary: Game videos may contain important information for game developers (e.g., bug descriptions). However, 21.4% of the games on the Steam platform receive more than 50 game videos in total, and up to a median of 13 hours of video runtime per day. Hence, manually watching through gameplay videos daily requires a considerable amount of resources. In addition, using a naïve keyword searching approach to identify bug videos only has a precision of 56.25%. Hence, a more advanced approach is needed to identify bug videos.

7.5 Determining the Likelihood that a Gameplay Video Showcases Game Bugs

In our preliminary study, we showed that naïve approaches do not perform well in identifying bug videos. To help game developers leverage the valuable information in gameplay videos such as bugs in the games, in this section, we propose an approach to improve the performance of the naïve keyword searching approach.

We train a random forest classifier to determine the likelihood that a game video showcases a bug of a game, based on the results of the keyword searching approach. We then evaluate the performance of our random forest classifier and use the classifier to understand which factors are correlated with a higher likelihood of a game video showcasing a game bug. By automatically determining the likelihood that a game video showcases a game bug, developers can watch game videos that have a high likelihood and avoid wasting time on videos that do not showcase a game bug. Figure 7.8 shows an overview of our approach.

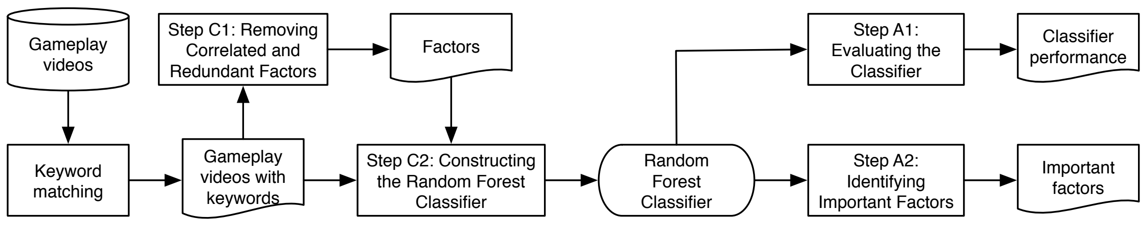
Gameplay videos
Keyword matching
Gameplay videos with keywords
Step C1: Removing Correlated and Redundant Factors
Factors
Step C2: Constructing the Random Forest Classifier
Random Forest Classifier
Step A1: Evaluating the Classifier
Classifier performance
Step A2: Identifying Important Factors
Important factors

Figure 7.8: Overview of the approach to determine the likelihood that a gameplay video showcases a game bug

7.5.1 Approach

To train the random forest classifier, we used factors that measure the contents of video posts and metadata of the YouTube video. Table 7.4 defines each factor and the rationale behind our choice of that factor. We used the identified keywords in our preliminary study (i.e., "bug", "glitch", "hack", "hacker", "cheat", "cheater") for features that concern keywords in this section. We did not include context factors (e.g., the game that the video is about) to ensure that our approach can be adapted by newly-released games. In addition, we did not include factors that are related to the account that uploads the video (e.g., the number of videos that were uploaded) to ensure that our approach works for videos from both newly-created and old accounts.

We built a random forest classifier to determine the likelihood that a game video showcases a game bug. We chose the random forest technique for our classifier as it is considered to be one of the best performing learning algorithms compared to SVM, boosted trees, Bayesian or logistic regression (Caruana and Niculescu-Mizil, 2006). We conducted two steps (C1, C2) to construct a random forest classifier, and two steps (A1, A2) to analyze the constructed classifier.

Table 7.4: Factors used in the random forest model

Factor	**Description (d) - Rationale (r)**
Video length	d: The length of the video in seconds. r: Bug videos could be shorter, as they may only show the bug instead of a full gameplay session.
Video post title length	d: The number of characters in the title of the video post. r: A longer title of the video post could contain a description of a game bug.
Video post description length	d: The number of characters in the description of the video post. r: A long description could contain stuffed keywords (Google, 2018), which are usually irrelevant to the content of the video and hence decrease the likelihood of the video showcasing a bug.
YouTube title length	d: The number of characters in the YouTube title. r: A longer YouTube title could contain a description of a game bug.
YouTube description length	d: The number of characters in the YouTube description. r: A long YouTube description could contain stuffed keywords (Google, 2018), which are usually irrelevant to the content of the video and hence decrease the likelihood of the video showcasing a bug.
# of YouTube tags	d: The number of YouTube tags. r: A video with a large number of YouTube tags could contain stuffed keywords (Google, 2018), which are usually irrelevant to the content of the video and hence decrease the likelihood of the video showcasing a bug.
# of keyword matches in video post title	d: The number of occurrences of the keywords in the title of the video post. r: A video post with more keyword matches in the title could have a higher likelihood of being a bug video.
# of keyword matches in video post description	d: The number of occurrences of the keywords in the description of the video post. r: A video post with more keyword matches in the description could have a higher likelihood of being a bug video.
# of keyword matches in YouTube title	d: The number of occurrences of the keywords in the YouTube title. r: A video with more keyword matches in the YouTube title could have a higher likelihood of being a bug video.
# of keyword matches in YouTube description	d: The number of occurrences of the keywords in the YouTube description. r: A video with more keyword matches in the YouTube description could have a higher likelihood of being a bug video.
# of keyword matches in YouTube tags	d: The number of occurrences of the keywords in the YouTube tags. r: A video with more keyword matches in YouTube tags could have a higher likelihood of being a bug video.

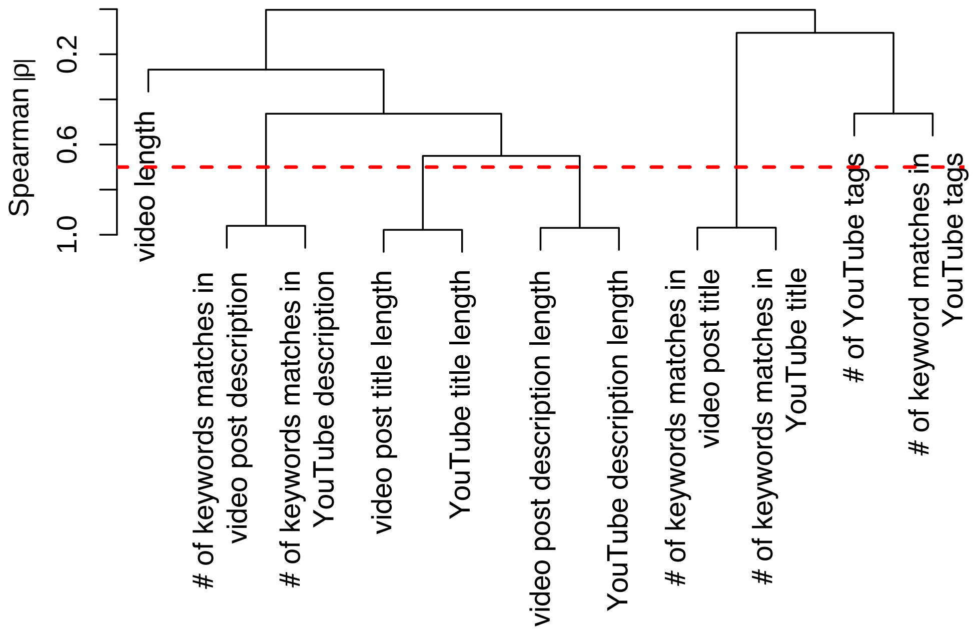

Figure 7.9: The hierarchically clustered Spearman $|\rho|$ values of factors

Step C1 - Removing Correlated and Redundant Factors

Correlated factors in a model can lead to the misinterpretation of factor importance (Tantithamthavorn and Hassan, 2018). To mitigate correlated factors, we used the Spearman rank correlation to calculate how strongly two factors are correlated. If two factors have a Spearman $|\rho| > 0.7$, we remove one of them from our model. We used Spearman rank correlation because we observed our data to be skewed, and Spearman rank correlation does not assume a normal distribution of the data. Figure 7.9 shows the hierarchically clustered Spearman $|\rho|$ values.

We found that the corresponding factors across YouTube and Steam (e.g., video post title length and YouTube title length) are all highly correlated. As stated in Section 7.2, although the video posts on the Steam platform are hosted on YouTube, Steam allows gamers to customize the title and description of the video posts. We found that most of the video post titles are the same as their YouTube titles. In fact, only 167,486 (1.0%) video posts have different titles compared to their YouTube titles. We manually looked into the videos with different titles for the video post and on YouTube, and identified the following common reasons:

- Steam filtered inappropriate words, while YouTube did not
- Gamers did not enter a title for the Steam video post
- Gamers added a prefix to their YouTube titles (e.g. their YouTube channel name)

In addition, only 192,998 (1.2%) videos have different descriptions on Steam and on YouTube. The most common differences are:

- Copyright disclaimers for the background music or other part of the videos in the YouTube description
- Stuffed keywords at the end of the YouTube description
- Descriptions in different languages

To ensure that our approach can be applied to game videos that are not on the Steam platform, we kept the factors from the YouTube metadata, and removed the video post title length, # of keywords matches in the video post title, video post description length and the # of keywords matches in the video post description from our classifier.

To further remove redundant factors (i.e., factors that can be explained by a combination of other factors), we performed a redundancy analysis using the `redun` function in the `Hmisc` package in R. We found that after removing the correlated factors, there are no redundant factors.

Step C2 - Constructing the Random Forest Classifier

As the classifier aims at improving the initial results of the keyword search, we used the manually labelled data in Section 7.4 that was used to evaluate the precision of the keyword searching approach to construct the random forest classifier. We used a default value of 3 for the `mtry` parameter (the number of variables that are randomly sampled as candidates at each split).

Step A1 - Evaluating the Classifier

To compare the performance of the classifier to the naïve keyword search approach, we used the out-of-sample bootstrap validation technique (Tantithamthavorn et al., 2017). The out-of-sample bootstrap validation technique consists of two steps:

1. For the given original dataset of size N, we randomly draw a bootstrap sample of size N with replacement.

2. We train the Random Forest classifier using the bootstrap sample, and test the classifier using the rows that are not drawn in the first step. As the bootstrap sample is drawn with replacement, on average 36.8% of the rows will not be drawn in the first step (Efron, 1983).

The out-of-sample bootstrap process was repeated 1,000 times, and the average out-of-sample precision and AUC was calculated.

As we aim at using the random forest classifier to provide a list of videos ranked by their likelihood of being a bug video, only using precision and AUC to evaluate the performance of the approach will ignore the order in which the bug videos are presented.

Hence, in addition to the out-of-sample bootstrap validation, we further quantify the performance of our approach using the following set of metrics:

Precision at n ($P@n$) (Larson, 2010) is commonly used to evaluate the performance of an information retrieval system. For our approach, we have $P@n = \frac{r}{n}$, where r is the number of actual bug videos at rank n.

Average Precision at n ($AP@n$) (Larson, 2010) is a variant of the well-known average precision measure. While precision at n ignores the positions of the result, average precision at n takes into account both the number of bug videos in the top n and the positions of those bug videos. The average precision at n is defined as:

$$AP@n = \frac{\sum_{i=1}^{n} rel(i) \times P@i}{NF}$$

Where $rel(i) = 1$ if the i^{th} video in the ranked list is a bug video, and $rel(i) = 0$ otherwise, and NF is the normalization factor. Although it is common to normalize by the total number of bug videos, here we use the number of correctly identified bug videos as the normalization factor, as proposed by Baeza-Yates et al. (1999). This normalization is called Average Precision at Seen Relevant Documents, and it avoids the problem in precision-oriented evaluation, where one may not have judged enough documents to know the recall. In the remainder of the chapter, we use $AP@n$ to refer to Average Precision at Seen Relevant Documents.

Mean Average Precision at n ($MAP@n$) (Larson, 2010) is the arithmetic mean of all average precision at n.

We selected 10 games from the Steam platform with the highest number of videos, and calculated the metrics for each of the games. We also selected 4 games that were not released on the Steam platform, and collected their game videos on the YouTube

Table 7.5: Games that are used for the evaluation of our approach

Title	**# of videos**	**# of videos identified by keyword matching**	**# of videos identified by random forest classifier**
Steam games			
Counter-Strike: Global Offensive	352,443	12,495	9,376
Dota 2	65,023	841	650
Garry's Mod	37,245	945	617
Arma 3	32,049	825	507
Grand Theft Auto V	31,127	2,310	1,386
Counter-Strike	26,377	728	537
DayZ	20,234	1,590	968
Rocket League	19,759	264	166
Counter-Strike: Source	17,691	1,046	574
ARK: Survival Evolved	17,080	639	198
Non-Steam games			
FIFA 16	5,701	396	161
FIFA 17	6,008	464	200
NHL 16	4,624	400	226
NHL 17	4,421	382	205

platform, to evaluate the generalizability of our approach. Table 7.5 shows the games we used for our evaluation.

For each of the games, we calculated the $P@n$ and $AP@n$ for $n = 100$. Hence, we manually verified whether 1,400 game videos showcase a bug. In addition, to simulate the process of developers using our proposed approach as they go through pages of results, with each page displaying 10 videos, we calculated the $AP@10$ of each page of results for each game. Lastly, we calculated the $MAP@n$ for $n = 100$, for the Steam games and non-Steam games respectively.

Step A2 - Identifying Important Factors

To understand the importance of each factor in a random forest classifier, we calculated the mean decrease accuracy (Louppe et al., 2013) for each factor. The mean decrease accuracy is commonly used in prior studies for measuring factor importance of random forest classifiers (Kabinna et al., 2018) and is calculated as follows: for each tree in the random forest, the error rate on the out-of-bag portion of the data is recorded, and compared with the error rate after permuting each factor. The bigger the difference is, the more important that factor is deemed. We used the `importance` function in the `RandomForest` package in `R` to calculate the mean decrease accuracy.

7.5.2 Results

Our random forest classifier achieves a precision that is 43% higher than the precision of the naïve keyword searching approach. The out-of-sample bootstrap validation shows that on average, the Random Forest classifier has an AUC of 0.84, and a precision of 0.80 on the manually labelled data, which is 43% higher than the precision of the naïve keyword searching approach (0.56, see Section 7.4).

Our random forest classifier achieves a mean average precision at 100 of 0.91. We manually checked the 100 gameplay videos with the highest likelihood of showcasing a game bug for the 14 games for evaluation. In total, we manually checked 1,400 gameplay videos. Table 7.6 shows the performance metrics for each of the evaluated games.

Table 7.6 shows that for both the 10 Steam games and the 4 non-Steam games that are used for evaluation, the classifier obtained a mean average precision at 100 of 0.91. For the *Garry's Mod* game, the *Rocket League* game and the *Counter-Strike: Source* game, we observed a relatively low $P@100$ (below 0.8). However, the $AP@100$ for these

Table 7.6: Result of the evaluation

Title	$P@100$	$AP@100$	$AP@10$									
			1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	91-100
Steam games												
Counter-Strike: Global Offensive	0.83	0.84	0.84	0.81	0.94	0.91	1.00	0.58	1.00	0.77	0.79	0.89
Dota 2	0.93	0.96	1.00	1.00	0.88	1.00	1.00	0.98	0.88	1.00	0.73	0.96
Garry's Mod	0.74	0.81	0.98	0.75	0.96	0.56	0.96	0.99	0.73	0.73	0.46	0.74
Arma 3	0.93	0.97	1.00	1.00	1.00	1.00	0.92	0.91	0.99	1.00	0.98	0.84
Grand Theft Auto V	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Counter-Strike	0.89	0.85	0.54	1.00	0.83	1.00	1.00	0.99	1.00	0.91	1.00	0.93
DayZ	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96	1.00	1.00
Rocket League	0.65	0.90	1.00	1.00	1.00	0.95	0.91	0.29	0.53	0.34	0.33	0.91
Counter-Strike: Source	0.72	0.83	0.75	0.96	0.98	0.96	0.82	1.00	0.81	0.92	0.50	0.11
ARK: Survival Evolved	0.92	0.97	1.00	1.00	1.00	0.88	1.00	0.75	1.00	1.00	0.57	0.93
$MAP@100$ for Steam games		0.91										
Non-Steam games												
FIFA 16	0.91	0.88	0.70	1.00	1.00	1.00	1.00	0.93	1.00	0.75	1.00	0.97
FIFA 17	0.96	0.96	0.99	1.00	1.00	0.88	1.00	1.00	1.00	1.00	0.96	0.96
NHL 16	0.90	0.99	1.00	1.00	1.00	1.00	0.95	1.00	1.00	1.00	0.55	0.46
NHL 17	0.89	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.76	0.44
$MAP@100$ for non-Steam games		0.91										

three games are all higher than their $P@100$ (all above 0.8, up to 0.25 higher than their $P@100$), suggesting that our random forest classifier is able to assign a higher likelihood of showcasing a bug to bug videos than to non-bug videos.

To further understand the false positive cases, we manually checked the 168 videos that were wrongly identified by our classifier as bug videos. The false positive cases were mainly introduced by the ambiguity of keywords, specifically, the keyword "hack". We noticed that in the *Counter-Strike* community (i.e., the *Counter-Strike* game, the *Counter-Strike: Source* game, and the *Counter-Strike: Global Offensive* game), based on context, the term "hack" can be used to describe gameplay that is so skilled that it looks like hacking (e.g., `http://youtu.be/RG78hWtY5Mo`). Another common reason was that gamers' usernames contain "hack" while the videos are not about hacking. In addition, we also noticed cases in which the videos were about hacked gamer accounts. We calculated that by excluding the keyword "hack" and "cheat", 23% of the

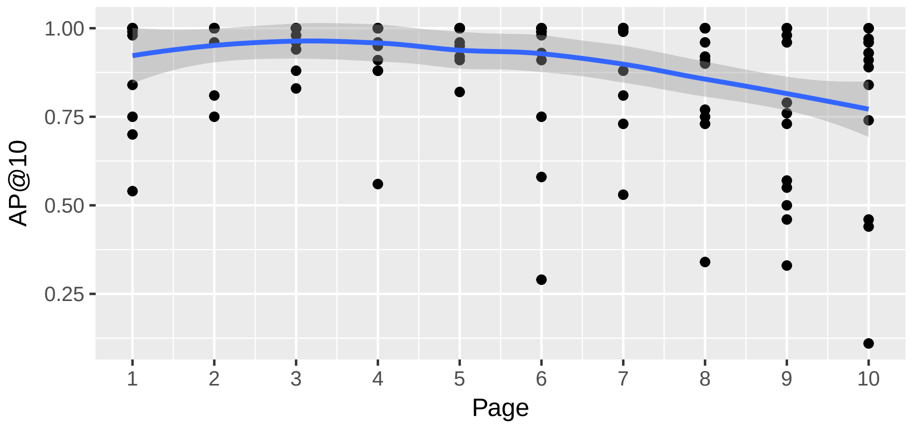

Figure 7.10: $AP@10$ against the page of results (with 10 videos per page). The blue lines are fitted using Local Polynomial Regression (Cleveland and Devlin, 1988). The grey areas show the approximate 95% confidence bands.

false positive cases would not be labelled as bug videos. Hence, developers can choose to exclude the keyword "hack" to further increase the precision of our approach, if desired. Other reasons for the false positive cases include videos about how to patch a bug, gamers using the word "buggy" to refer to motor vehicles, gamers' usernames containing keywords, and keyword stuffing.

The average precision at 10 of each page of results remains high, even after already having seen 10 pages of results. Figure 7.10 shows the relationship between $AP@10$ and the page of results, with each page containing 10 videos. Even for page 10, the classifier obtains an average $AP@10$ of 0.80, and a $MAP@10$ of 0.92. Although more study may be needed to evaluate how well the classifier performs after 10 pages, the classifier still has a high performance even after having seen 10 pages of results already.

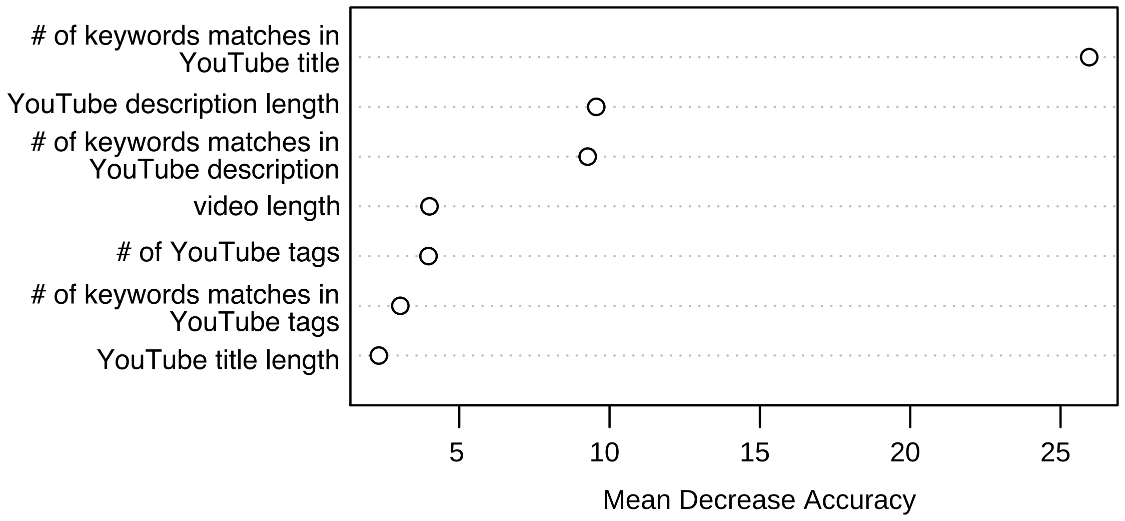

Figure 7.11: The mean decrease accuracy for each factor in the classifier

The most important factor for identifying bug videos is having a keyword in the YouTube title. Figure 7.11 shows the mean decrease accuracy of each factor that is used in our random forest classifier.

As Figure 7.11 shows, the number of keyword matches in YouTube title has the highest mean decrease accuracy, with the length of the YouTube description, and the number of keyword matches in the YouTube description being the second and the third respectively. Hence, the number of keyword matches in the YouTube title is the most important factor for identifying bug videos.

To understand how changes in the three most important factors' values affect the likelihood of a game video being a bug video, we plotted the predicted likelihood against the three aforementioned factors. Figure 7.12 shows the impact of the three most important factors on the likelihood of a game video being a bug video.

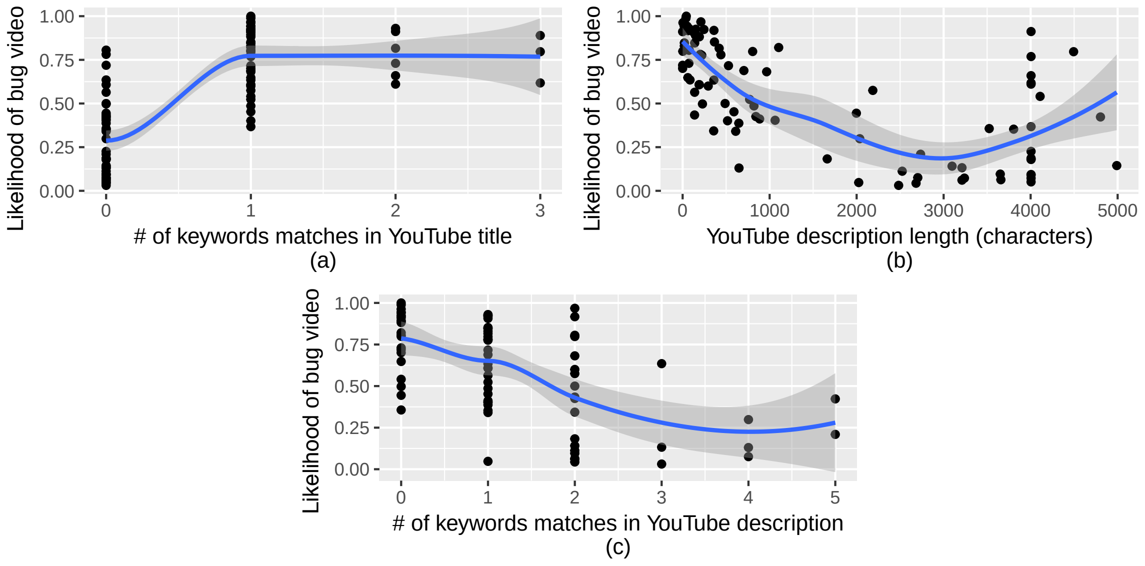

Figure 7.12: The predicted likelihood of bug videos against the three most important factors for deciding the likelihood of a video being a bug video. The blue lines are fitted using Local Polynomial Regression (Cleveland and Devlin, 1988). The grey areas show the approximate 95% confidence bands.

For the number of keyword matches in the YouTube title, Figure 7.12(a) shows that the likelihood of the video showcasing a bug is higher when there is at least one occurrence of a keyword, but the likelihood does not go higher as the occurrence increases. For the number of keywords matches in the YouTube description, Figure 7.12(c) shows that the likelihood decreases as the occurrence of keywords in YouTube description increases. A possible explanation is that, as stated in Section 7.4, people may stuff irrelevant keywords in the videos' YouTube descriptions as an attempt of search engine optimization. In addition, Figure 7.12(b) shows that the likelihood of a video being a bug video is higher when the length is either very short or very long.

Summary: The proposed random forest classifier achieves a mean average precision at 100 of 0.91, and a precision (0.80) that is 43% higher than the naïve keyword searching approach (0.56). Videos with at least one occurrence of a keyword (i.e., "bug", "glitch", "hack", "hacker", "cheat", "cheater") in their YouTube title, a shorter YouTube description, and lower occurrence of keywords in their YouTube description, are more likely to be bug videos.

7.6 Related Work

In this section, we discuss prior work that is related to our study. The contribution of our study in comparison to prior work is that to the best of our knowledge, we are the first to propose an approach for identifying game videos that showcase a game bug.

7.6.1 Studies on Gameplay Videos

Lewis et al. (2010) explored the software engineering aspect of gameplay videos. Lewis et al. summarized a taxonomy of video game bugs by observing patterns from bug videos on YouTube. They identified YouTube's bug videos as *"a rich resource ... provide a startling amount of coverage; far more than any single research group could ever hope to expose personally"*.

Most of the studies on gameplay videos focus on using gameplay videos to automatically generate game levels. Guzdial and Riedl (2016) designed an unsupervised machine learning process that can automatically generate full video game levels from gameplay videos, using K-Means and probabilistic graphical models. Summerville et al. (2016) proposed a machine learning technique that uses Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs) to generate levels based on latent play styles learnt from the Super Mario Bros gameplay videos.

7.6.2 Studies on Improving Bug Report Quality

Several studies have focused on the quality of bug reports. Zimmermann et al. (2010) conducted a survey among the community of Apache, Eclipse and Mozilla to understand the characteristics of high quality bug reports, and revealed the mismatch of information between what developers need and what bug reporters provide. The study highlights that for developers, the most needed information is the steps to reproduce, which is considered by users as difficult to provide. In addition, the most severe problem is not wrong information, but absent information. The study also highlights the potential value of video bug reports, using the "Best of Bugzilla" bug report, Eclipse bug 113206 as an example. This bug report includes a video to demonstrate the complicated steps to reproduce the bug. Hooimeijer and Weimer (2007) analyzed over 27,000 bug reports for Mozilla Firefox to model the bug report quality. The analysis showed that attachment count and comment count have the highest impact on the quality of a bug report. Linstead and Baldi (2009) proposed a new measure of bug report quality called latent topic coherence, by modelling Gnome bug reports with Latent Dirichlet Allocation (LDA). Zimmermann et al. (2009) proposed four broad directions for enhancing the current bug tracking systems to collect bug reports with higher quality, one of which suggests the bug reporting tool to integrate capture/replay functionality or a screenshot.

7.7 Threats to Validity

This section presents the threats to the validity of our findings.

7.7.1 Internal Validity

A threat to the validity of our findings is that we only collected videos that are explicitly linked to specific games, on both the Steam platform, and YouTube. This constraint ensures that the number of videos of a game is an accurate lower bound. However, as it is not mandatory for gamers to link videos to games in either the Steam platform or on YouTube, there may exist more videos related to studied games that were not collected.

In both Section 7.4 and Section 7.5, we manually verified whether a gameplay video is a bug video, for a total of 1,496 gameplay videos. While this process was manual, it was straightforward. We would like to emphasize that our model is necessary to reduce the required time to identify bug videos, not to reduce the difficulty of the identification process. Because of the low degree of difficulty of the identification, we did not cross-validate the identification process.

In this chapter, we built a universal model to predict the likelihood that a gameplay video showcases a game bug across all games. However, the gamer community of different games may have different patterns when posting bug videos. In this case, game-specific models may outperform a universal model. In practice, newly-released games may not have enough historical data to train a game-specific model. Nonetheless, our approach can be adopted for building game-specific models as well.

7.7.2 External Validity

In our preliminary study (Section 7.4), we focused on the gameplay videos on the Steam platform. The findings of our preliminary study may not be generalizable to other games on different distribution platforms. However, as stated in Section 7.2, Steam is the largest online distribution platform for PC games. Hence, the games on the Steam platform are representative for a large number of games. Future studies are necessary to investigate how our results apply to games from other platforms, such as the Xbox.

In Section 7.5, we evaluated the generalizability of our approach using four games that are not on the Steam platform. Future studies are needed to examine the generalizability of the approach on a larger number of games.

7.8 Chapter Summary

The user-perceived quality is crucial to the success of a game. Hence, it is necessary to efficiently and effectively deal with bugs in a game. Common practices of collecting bug reports are either reporter-dependent (hence hard to automate), lack valid reproduction steps, or are intrusive with privacy concerns. Although in recent years gameplay videos have become popular in the gaming community to showcase the problems of a game to other gamers, there has been no studies of how developers can use such gameplay videos as a source for bug reports.

In this chapter, we explored the practicability of using gameplay videos that are available online as a supplemental source of bug reports for games. We firstly conducted a preliminary study on gameplay videos on the Steam platform, and found that naïve approaches to identify bug videos, such as keyword searching, are inefficient and imprecise. In particular, the naïve keyword searching approach only has a precision of 56.25%. We then proposed an approach that uses the metadata of gameplay videos to train a random forest classifier, to determine the likelihood that a gameplay video showcases a game bug. We evaluated our approach on 10 Steam games and 4 non-Steam games. Overall, the approach achieves both a mean average precision at 10 and a mean average precision at 100 of 0.91, and shows a good generalizability. We also identified the impact of different factors on the likelihood of a video related to bugs. Our study makes the following contributions:

- A demonstration of the value of mining gamer-produced audiovisual content for software engineering.
- The first to showcase that it is practical to automatically identify bug videos with high precision.
- An analysis of the characteristics of bug videos.
- A labelled dataset of 1,400 bug videos[7] to encourage further studies in this important research direction.

Our contributions can help developers better utilize gameplay videos. In particular, our approach and findings allow game developers to quickly identify bug videos of their own games.

[7]https://github.com/SAILResearch/replication-bug_videos

CHAPTER 8

Conclusion and Future Work

THE gaming industry is a multi-billion dollar industry with a user base that is extremely difficult to satisfy, making developing a successful game challenging. Prior studies on mining mobile app stores have yielded important insights and suggestions to help mobile app developers. However, mining online distribution platforms for games to assist and guide game development has not been explored in literature. As prior work has shown that developing a game is very different from developing other types of software, knowledge derived from mining mobile app stores may not be directly applicable to game development. In this Ph.D. thesis, we mined data from a dominant online distribution platform for games, the Steam platform, to provide practical suggestions and techniques for developers to produce games with an improved user-perceived quality. In particular, we focused on the following four aspects of online distribution for games: urgent updates, the early access model, user reviews, and user-recorded gameplay videos on the Steam platform. The findings of this thesis offer insight into the software practices of game developers, provide useful guidelines for game development, and inspire future longitude studies in software engineering for games.

8.1 Thesis Contributions

The goal of this thesis is to highlight the promising outcome of mining online distribution platforms for games to provide practical suggestions for game developers. In our literature survey, we showed that extensive research has been conducted in the field of mining mobile app stores. However, prior studies confirmed the large differences between developing a traditional non-game and a game software system, raising threats to directly applying knowledges from prior mobile app store research to game engineering. Below, I summarize the main contributions of this thesis around the four studied aspects of online distribution for games:

1. **The choice of update strategy is associated with the proportion of 0-day updates.** Urgent updates are usually released in a state of emergency, causing unnecessary stress on developers. The stress of these so-called "fire-fighting conditions" can not only lead to inefficient problem solving, but also introduce changes that can easily create new problems (Bohn, 2000). In this thesis, We empirically studied urgent updates of games on the Steam platform to understand the causes behind urgent updates. We observed that games that release frequently also release a higher proportion of 0-day updates than games that

use a traditional build-up candidate update strategy. Our findings are consistent with the findings of Souza et al. (2015), who observed that releasing frequently leads to a higher proportion of patches that must be reverted.

2. **The early access model helps developers elicite early feedback and more positive reviews to attract additional new players.** In order to get a better understanding of the impact and limitations of the early access model, we conducted an in-depth empirical study on early access games on the Steam platform. We observed a correlation between early access games and a higher positive review rate. While the early access model is not a fix for low-quality games, the early access model appears to be a valuable tool for developers that want to improve their games by interacting with their players.

3. **Although negative reviews contain more valuable information for developers, the portion of useful information in positive reviews should not be ignored by developers and researchers.** The competition within the gaming industry, and the hard-to-please user base has made the quality of games an increasingly important issue. As game reviews are a direct reflection of user concerns, a better understanding of reviews can help developers produce games with an improved user-perceived quality. In this thesis, we performed an empirical study on the reviews of games on the Steam platform. We observed that 29% of the positive reviews discuss cons of the games, and 7% of the positive reviews report bugs in the games. Moreover, positive reviews contain a higher portion of pros of the games, and a slightly higher portion of suggestions, than negative reviews. Hence, developers and researchers should not dismiss the information that can be extracted from positive reviews.

4. **User-recorded gameplay videos that are available online can be used as a supplemental source of bug reports for games.** In recent years, gameplay videos have become popular in the gaming community. The high amount of videos around game bugs opens a new opportunity for developers to collect intuitive information of bugs. In this thesis, we proposed an approach that uses the metadata of gameplay videos to train a random forest classifier, to determine the likelihood that a gameplay video showcases a game bug. Our approach achieves a mean average precision at 100 of 0.91, and shows a good generalizability.

8.2 Future Research Directions

The results of our empirical studies highlighted the promising outcome of mining online distribution platforms for games to provide practical suggestions for game developers. In addition to the results stated in this thesis, there are other aspects of online distribution that we think should be examined by future research efforts.

8.2.1 The impact of rapid updating of games on well-established software engineering practices

Prior work (Khomh et al., 2012, 2015; da Costa et al., 2016) on update strategies focused mostly on the Mozilla Firefox project, in which the update strategy changed from traditional build-up candidate updates to frequent updates (i.e., every six weeks). In this thesis, we showed that most games update much more frequently than once every six weeks (see Chapter 4), a phenomenon that was recently observed for mobile apps (McIlroy et al., 2016a). The unique distribution mechanism (e.g. online store) of games and mobile apps allows developers to release updates for their software at an increasingly rapid pace. Future research efforts need to carefully reconsider how such rapid pace of updating software influences our well-established understandings of software engineering practices. For example, how should the practice of requirement engineering and testing be adjusted to cope with such rapid update pace for games.

8.2.2 A deep understanding is needed for the relation between using the early access model and the satisfaction of both players and developers.

In this thesis, we observed a correlation between early access games and a higher positive review rate (see Chapter 5). One possible explanation for this correlation is that players who buy EAGs are friendlier towards developers. Another explanation is that developers that use the early access model are good at keeping their players satisfied. Future studies should use methods such as developer surveys, user studies, and controlled experiments to examine in more depth the causality between using the early access model and the satisfaction of both players and developers.

8.2.3 The early access model for non-game software.

In this thesis, we investigated the early access model for games and provided game developers with suggestions on how to leverage the early access model. Some traditional software has also adopted similar development models, for example, the "Office Insider" program for Microsoft Office (Foley, 2015). Future studies should verify if such models for non-game software share similar characteristics as the early access model.

8.2.4 Adjusting existing automated techniques for analyzing mobile app reviews so they can be applied successfully to game reviews.

Throughout our study of reviews for games on the Steam platform, we observed that in several aspects, game reviews are different from mobile app reviews (see Chapter 6). In addition, we noticed that reviews for games tend to be sarcastic (e.g., "Went to washroom for one minute and lost the game. 10/10."), posing threats to directly applying automated sentiment analysis or information retrieval techniques for analyzing mobile app reviews on game reviews. Hence, in this thesis we conducted manual categorization of statistically representative samples of reviews to ensure the quality of our findings. Future studies should investigate further how to adjust existing automated techniques for analyzing free form reviews of games and mobile apps.

8.2.5 Revisiting prior work on mobile app reviews to take missing factors into account.

Prior research on mobile app reviews only had access to the number of owners of an app, while our study analyzed both the number of owners and the number of active users (players) of games, and observed that the number of players has a stronger relation with the number of reviews received per day. In addition, prior mobile app studies that do study paid apps, tend to ignore the impact of discounts. These studies of mobile app reviews should be revisited, as we observed that a sales event has the strongest relation with an increase in the number of received game reviews.

8.2.6 Comparing mobile games with PC games.

Our studies focused on mining the Steam platform, which is an online distribution platform dedicated for PC games. Prior studies on mobile games (Alves et al., 2008; Duh et al., 2008; Korhonen et al., 2009; Korhonen and Koivisto, 2007) show that developing mobile games has its special software and hardware challenges and peculiarities, compared to developing PC games. Future studies should investigate whether the findings and suggestions in our thesis hold for mobile game development.

8.2.7 Surveying and interviewing game developers to evaluate the suggestions from this thesis.

In this thesis, we studied four aspects of online distribution for games to provide valuable insights and practical suggestions. Our research received widespread coverage from several prestigious media outlets in the gaming industry, including PC Gamer (Wood, 2018), Kotaku (Booker, 2017; Plunkett, 2018), and Gamasutra (Kidwell, 2018). These outlets receive up to 50 million visitors per month and are the leading venues in the gaming community. In addition, our publications have been read for more than 18,000 times on ResearchGate alone. The overwhelming attention and positive feedback that we received from the gaming community showcase the relevance of our studies to this community. Nevertheless, future studies should consider other more structured avenues (e.g., interview and surveys) to gather feedback from the gaming community about our work and our suggestions.

Bibliography

Afic, R. (2015). How to report a game bug. https://boards.na.leagueoflegends.com/en/c/bug-report/3mQGBEjA-how-to-report-a-game-bug. (last visited: Oct 16, 2018).

Ahmed, F., Zia, M., Mahmood, H., and Al Kobaisi, S. (2017). Open source computer game application: An empirical analysis of quality concerns. *Entertainment Computing*, 21:1–10.

Al-Ani, B., Trainer, E., Ripley, R., Sarma, A., Van Der Hoek, A., and Redmiles, D. (2008). Continuous coordination within the context of cooperative and human aspects of software engineering. In *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, pages 1–4. ACM.

Alatalo, T., Kuusela, E., Puuperä, R., and Ojala, T. (2013). Comparative api complexity analysis of two platforms for networked multiplayer games using a reference game.

In *Proceedings of the 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change*, pages 44–50. IEEE Press.

Albassam, E. and Gomaa, H. (2013). Applying software product lines to multiplatform video games. In *Proceedings of the 3rd International Workshop on Games and Software Engineering (GAS)*, pages 1–7. IEEE.

Alden (2017). Steamworks Partner Program. `https://partner.steamgames.com/steamdirect`. (last visited: Oct 16, 2018).

Alexa (2018). Youtube.com traffic, demographics and competitors - alexa. `https://www.alexa.com/siteinfo/youtube.com`. (last visited: Oct 16, 2018).

Allen, P. (2016). Is Steam's Early Access a ticking time bomb? `http://www.gamasutra.com/blogs/PaulAllen/20160211/265598/Is_Steams_Early_Access_a_ticking_time_bomb.php`. (last visited: Oct 16, 2018).

Alves, V., Câmara, T., and Alves, C. (2008). Experiences with mobile games product line development at meantime. In *Proceedings of the 12th International Software Product Line Conference (SPLC'08)*, pages 287–296. IEEE.

Ampatzoglou, A. and Stamelos, I. (2010). Software engineering research for computer games: A systematic review. *Information and Software Technology*, 52(9):888–901.

Arora, A., Caulkins, J. P., and Telang, R. (2006). Research note: Sell first, fix later: Impact of patching on software quality. *Management Science*, 52(3):465–471.

Arora, A., Krishnan, R., Telang, R., and Yang, Y. (2010). An empirical analysis of software vendors' patch release behavior: Impact of vulnerability disclosure. *Information Systems Research*, 21(1):115–132.

Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval*, volume 463. ACM press New York.

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

Bavota, G., Linares-Vasquez, M., Bernal-Cardenas, C. E., Di Penta, M., Oliveto, R., and Poshyvanyk, D. (2015). The impact of api change-and fault-proneness on the user ratings of android apps. *IEEE Transactions on Software Engineering*, 41(4):384–407.

Bécares, J. H., Valero, L. C., and Martín, P. P. G. (2017). An approach to automated videogame beta testing. *Entertainment Computing*, 18:79–92.

Becker, R., Chernihov, Y., Shavitt, Y., and Zilberman, N. (2012). An analysis of the Steam community network evolution. In *Proceedings of the 27th Convention of Electrical & Electronics Engineers in Israel (IEEEI)*, pages 1–5. IEEE.

Bierzerkers (2016). DEVBLOG 5. `http://archive.is/XHOjf`. (last visited: Oct 16, 2018).

Blackburn, J., Kourtellis, N., Skvoretz, J., Ripeanu, M., and Iamnitchi, A. (2014). Cheating in online games: A social network perspective. *ACM Transactions on Internet Technology (TOIT)*, 13(3):9.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Bohn, R. (2000). Stop fighting fires. *Harvard Business Review*, 78(4):82–91.

Booker, L. (2017). Research Paper On Steam Early Access Reveals 5 Lessons For Developers. https://www.kotaku.com.au/2017/06/research-paper-on-steam-early-access-reveals-5-lessons-for-developers/. (last visited: Oct 16, 2018).

Cards and Castles (2015). Free To Play Coming Soon! http://store.steampowered.com/news/?appids=360730&headlines=1. (last visited: Oct 16, 2018).

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM.

Castro, M., Costa, M., and Martin, J.-P. (2008). Better bug reporting with better privacy. *ACM SIGARCH Computer Architecture News*, 36(1):319–328.

Chambers, C., Feng, W.-c., Sahu, S., and Saha, D. (2005). Measurement-based characterization of a collection of on-line games. In *Proceedings of the 5th Conference on Internet Measurement*. USENIX Association.

Chambers, J. M. and Hastie, T. J. (1991). *Statistical models in S*. CRC Press, Inc.

Chen, N., Lin, J., Hoi, S. C., Xiao, X., and Zhang, B. (2014). Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, pages 767–778. ACM.

Cheung, G. K., Zimmermann, T., and Nagappan, N. (2014). The first hour experience: How the initial play can engage (or lose) new players. In *Proceedings of the First*

Annual Symposium on Computer-human Interaction in Play (CHI PLAY), pages 57–66, New York, NY, USA. ACM.

Ciurumelea, A., Schaufelbühl, A., Panichella, S., and Gall, H. C. (2017). Analyzing reviews and code of mobile apps for better release planning. In *Proceedings of the 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 91–102. IEEE.

Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American statistical association*, 83(403):596–610.

Clifford, C. (2014). Crowdfunding Generates More Than $60,000 an Hour (Infographic). http://www.entrepreneur.com/article/234051?newsletter=true. (last visited: Oct 16, 2018).

Cobbett, R. (2017). From shareware superstars to the Steam gold rush: How indie conquered the PC. http://www.pcgamer.com/from-shareware-superstars-to-the-steam-gold-rush-how-indie-conquered-the-pc/. (last visited: Oct 16, 2018).

Coleman, M. and Liau, T. L. (1975). A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

CPx (2015). Pool Nation FX - Lite NOW AVAILABLE. http://store.steampowered.com/news/?appids=314000&headlines=1. (last visited: Oct 16, 2018).

da Costa, D. A., McIntosh, S., Kulesza, U., and Hassan, A. E. (2016). The impact of switching to a rapid release cycle on the integration delay of addressed issues: an empirical study of the Mozilla Firefox project. In *Proceedings of the 13th International Workshop on Mining Software Repositories*, pages 374–385. ACM.

Damodaran, L. (1996). User involvement in the systems design process-a practical guide for users. *Behaviour & information technology*, 15(6):363–377.

Daneva, M. (2017). Striving for balance: A look at gameplay requirements of massively multiplayer online role-playing games. *Journal of Systems and Software*, 134:54–75.

Di Sorbo, A., Panichella, S., Alexandru, C. V., Shimagaki, J., Visaggio, C. A., Canfora, G., and Gall, H. C. (2016). What would users change in my app? summarizing app reviews for recommending software changes. In *Proceedings of the 24th International Symposium on Foundations of Software Engineering*, pages 499–510. ACM.

Djundik, P. and Benjamins, M. (2016). Steam DB - Steam Database. `https://steamdb.info/`. (last visited: Oct 16, 2018).

Dragert, C., Kienzle, J., and Verbrugge, C. (2011). Toward high-level reuse of statechart-based ai in computer games. In *Proceedings of the 1st International Workshop on Games and Software Engineering*, pages 25–28. ACM.

Dragert, C., Kienzle, J., and Verbrugge, C. (2012). Reusable components for artificial intelligence in computer games. In *Proceedings of the Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques*, pages 35–41. IEEE Press.

Drescher, C., Wallner, G., Kriglstein, S., Sifa, R., Drachen, A., and Pohl, M. (2018). What moves players?: Visual data exploration of twitter and gameplay data. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 560. ACM.

Duh, H. B.-L., Chen, V. H. H., and Tan, C. B. (2008). Playing different games on different phones: an empirical study on mobile gaming. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 391–394. ACM.

Efron, B. (1983). Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American statistical association*, 78(382):316–331.

Electronic Arts Inc. (2018). Origin. https://www.origin.com. (last visited: Oct 16, 2018).

EuroGamer (2014). DayZ week-one sales rocket past 400,000. http://www.eurogamer.net/articles/2014-01-02-dayz-week-one-sales-rocket-past-400-000. (last visited: Oct 16, 2018).

Fine, D. (2016). Double Fine - What is Double Fine? http://www.doublefine.com/about/. (last visited: Oct 16, 2018).

Foley, M. J. (2015). Microsoft launches new Office Insider test program. http://www.zdnet.com/article/microsoft-launches-new-office-insider-test-program/. (last visited: Oct 16, 2018).

Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., and Sadeh, N. (2013). Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*, pages 1276–1284. ACM.

Gagné, A. R., Seif El-Nasr, M., and Shaw, C. D. (2012). Analysis of telemetry data from a real-time strategy game: A case study. *Computers in Entertainment (CIE)*, 10(1):2.

Gallivan, M. J. and Keil, M. (2003). The user–developer communication process: a critical case study. *Information Systems Journal*, 13(1):37–68.

Galyonkin, S. (2018). SteamSpy - All the data and stats about Steam games. http://steamspy.com/. (last visited: Oct 16, 2018).

Gao, C., Zeng, J., Lyu, M. R., and King, I. (2018). Online app review analysis for identifying emerging issues. In *Proceedings of the 40th International Conference on Software Engineering*, pages 48–58. ACM.

Genc-Nayebi, N. and Abran, A. (2017). A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software (JSS)*, 125:207–219.

Google (2017). How to use the Play Console. https://support.google.com/googleplay/android-developer/answer/6112435?hl=en. (last visited: Oct 16, 2018).

Google (2018). Irrelevant keywords - search console help. https://support.google.com/webmasters/answer/66358. (last visited: Oct 16, 2018).

Graham, T. N. (2010). Five grand challenges in the engineering of networked digital games. In *Proceedings of the Design and Engineering of Game-like Virtual and Multimodal Environments*.

Graham, T. N. and Roberts, W. (2006). Toward quality-driven development of 3d computer games. In *Proceedings of the International Workshop on Design, Specification, and Verification of Interactive Systems*, pages 248–261. Springer.

Grano, G., Di Sorbo, A., Mercaldo, F., Visaggio, C. A., Canfora, G., and Panichella, S. (2017). Android apps and user feedback: a dataset for software evolution and quality improvement. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics*, pages 8–11. ACM.

Gray, J. (2016). Steam Charts - Tracking What's Played. http://steamcharts.com/. (last visited: Oct 16, 2018).

Greg, D. (2014). Double Fine - Action Forums. https://archive.is/vFdqw. (last visited: Oct 16, 2018).

Gu, X. and Kim, S. (2015). What parts of your apps are loved by users? In *Proceedings of the 30th International Conference on Automated Software Engineering (ASE)*, pages 760–770. IEEE.

Guana, V., Stroulia, E., and Nguyen, V. (2015). Building a game engine: A tale of modern model-driven engineering. In *Proceedings of the 4th International Workshop on Games and Software Engineering (GAS)*, pages 15–21. IEEE.

Guzdial, M. and Riedl, M. (2016). Game level generation from gameplay videos. In *Proceedings of the Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Guzman, E., El-Haliby, M., and Bruegge, B. (2015). Ensemble methods for app review classification: An approach for software evolution (n). In *Proceedings of the 30th International Conference on Automated Software Engineering (ASE)*, pages 771–776. IEEE.

Habakuk, C. (2017). Bug reporting - eve community. https://community.eveonline.com/support/test-servers/bug-reporting/. (last visited: Oct 16, 2018).

Hall, R. J. (2011). Software engineering challenges of multi-player outdoor smartphone games. In *Proceedings of the 1st International Workshop on Games and Software Engineering*, pages 52–55. ACM.

Harman, M., Jia, Y., and Zhang, Y. (2012). App store mining and analysis: Msr for app stores. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 108–111. IEEE.

Harpstead, E., Zimmermann, T., Nagapan, N., Guajardo, J. J., Cooper, R., Solberg, T., and Greenawalt, D. (2015). What drives people: Creating engagement profiles of players from game log data. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, pages 369–379. ACM.

Hassan, S., Shang, W., and Hassan, A. E. (2017a). An empirical study of emergency updates for top android mobile apps. *Empirical Software Engineering*, 22(1):505–546.

Hassan, S., Tantithamthavorn, C., Bezemer, C.-P., and Hassan, A. E. (2017b). Studying the dialogue between users and developers of free apps in the google play store. *Empirical Software Engineering*, pages 1–38.

Hastjarjanto, T., Jeuring, J., and Leather, S. (2013). A dsl for describing the artificial intelligence in real-time video games. In *Proceedings of the 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change*, pages 8–14. IEEE Press.

Hooimeijer, P. and Weimer, W. (2007). Modeling bug report quality. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 34–43. ACM.

Hoon, L., Vasa, R., Schneider, J.-G., and Mouzakis, K. (2012). A preliminary analysis of vocabulary in mobile app user reviews. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, pages 245–248. ACM.

Hu, H., Bezemer, C.-P., and Hassan, A. E. (2016). Studying the consistency of star ratings and the complaints in 1 & 2-star user reviews for top free cross-platform android and ios apps. *Empirical Software Engineering*, pages 1–34.

Hu, H., Wang, S., Bezemer, C.-P., and Hassan, A. E. (2018). Studying the consistency of star ratings and reviews of popular free hybrid android and ios apps. *Empirical Software Engineering*, pages 1–26.

Huang, J., Zimmermann, T., Nagapan, N., Harrison, C., and Phillips, B. C. (2013). Mastering the art of war: How patterns of gameplay influence skill in Halo. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 695–704. ACM.

Iacob, C. and Harrison, R. (2013). Retrieving and analyzing mobile apps feature requests from online reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR)*, pages 41–44. IEEE.

Inc., A. (2017). Choosing a Membership. https://developer.apple.com/support/compare-memberships/. (last visited: Oct 16, 2018).

Jacobs, M. and Sihvonen, T. (2011). In perpetual beta? on the participatory design of facebook games. In *Proceedings of the Think Design Play: The Fifth International Conference of the Digital Research Association (DIGRA)*.

Johnson, J. (1999). Turning chaos into success. *Software Magazine*, 19(3):30–34.

Johnson, J. (2000). Chaos in the new millennium: The ghost of christmas future. *The Standish Group, West Yarmouth, MA*.

Kabinna, S., Bezemer, C.-P., Shang, W., Syer, M. D., and Hassan, A. E. (2018). Examining the stability of logging statements. *Empirical Software Engineering*, 23(1):290–333.

Kasurinen, J., Palacin-Silva, M., and Vanhala, E. (2017). What concerns game developers? a study on game development processes, sustainability and metrics. In *Proceedings of the 8th Workshop on Emerging Trends in Software Metrics (WETSoM)*, pages 15–21. IEEE.

Kerzazi, N. and Adams, B. (2016). Botched releases: Do we need to roll back? Empirical study on a commercial web app. In *Proceedings of the 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 574–583. IEEE.

Khalid, H., Nagappan, M., Shihab, E., and Hassan, A. E. (2014). Prioritizing the devices to test your app on: A case study of android game apps. In *Proceedings of the International Symposium on Foundations of Software Engineering*, pages 610–620. ACM.

Khalid, H., Shihab, E., Nagappan, M., and Hassan, A. E. (2015). What do mobile app users complain about? *IEEE Software*, 32(3):70–77.

Khomh, F., Adams, B., Dhaliwal, T., and Zou, Y. (2015). Understanding the impact of rapid releases on software quality. *Empirical Software Engineering*, 20(2):336–373.

Khomh, F., Dhaliwal, T., Zou, Y., and Adams, B. (2012). Do faster releases improve software quality? an empirical case study of mozilla firefox. In *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR)*, pages 179–188. IEEE.

Kidwell, E. (2018). Steam reviews study suggests "bad design", not bugs, irks players most. http://www.gamasutra.com/view/news/317707/Steam_reviews_study_suggests_bad_design_not_bugs_irks_players_most.php. (last visited: Oct 16, 2018).

Kim, B. C., Chen, P.-Y., and Mukhopadhyay, T. (2011). The effect of liability and patch release on software security: The monopoly case. *Production and Operations Management*, 20(4):603–617.

Kincaid, J. (1975). *Derivation of New Readability Formulas: (automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel*. Research Branch report 8–75. Chief of Naval Technical Training, Naval Air Station Memphis.

Köhler, B., Haladjian, J., Simeonova, B., and Ismailović, D. (2012). Feedback in low vs. high fidelity visuals for game prototypes. In *Proceedings of the Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques*, pages 42–47. IEEE Press.

Koleva, B., Tolmie, P., Brundell, P., Benford, S., and Rennick Egglestone, S. (2015). From front-end to back-end and everything in-between: work practice in game development. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, pages 141–150. ACM.

Korhonen, H. and Koivisto, E. M. (2007). Playability heuristics for mobile multi-player games. In *Proceedings of the 2nd International Conference on Digital Interactive Media in Entertainment and Arts*, pages 28–35. ACM.

Korhonen, H., Paavilainen, J., and Saarenpää, H. (2009). Expert review method in game evaluations: comparison of two playability heuristic sets. In *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, pages 74–81. ACM.

Kujala, S. (2003). User involvement: a review of the benefits and challenges. *Behaviour & information technology*, 22(1):1–16.

LadyAijou (2014). About the early access - Edge of Space General Discussions. `http://steamcommunity.com/app/238240/discussions/0/45350790838400003/#c45350790880063101`. (last visited: Oct 16, 2018).

LadyAijou (2015). ETA on getting out of Early Access? - Edge of Space General Discussions. http://steamcommunity.com/app/238240/discussions/0/419738204338030521/#c594820656453333967. (last visited: Oct 16, 2018).

Larson, R. R. (2010). Introduction to information retrieval. *Journal of the American Society for Information Science and Technology*, 61(4):852–853.

LeBreton, J. (2014). JP LeBreton's twitter. https://twitter.com/vectorpoem/status/536933544472240128. (last visited: Oct 16, 2018).

Lesensmer (2013). Hotfix - League of Legends Wiki - wikia. http://leagueoflegends.wikia.com/wiki/Hotfix. (last visited: Oct 16, 2018).

Lewis, C. and Whitehead, J. (2011a). Repairing games at runtime or, how we learned to stop worrying and love emergence. *IEEE software*, 28(5):53–59.

Lewis, C. and Whitehead, J. (2011b). The whats and the whys of games and software engineering. In *Proceedings of the 1st international workshop on games and software engineering*, pages 1–4. ACM.

Lewis, C., Whitehead, J., and Wardrip-Fruin, N. (2010). What went wrong: a taxonomy of video game bugs. In *Proceedings of the 5th International Conference on the Foundations of Digital Games (FDG)*, pages 108–115. ACM.

Leys, C., Ley, C., Klein, O., Bernard, P., and Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766.

Lin, D., Bezemer, C.-P., and Hassan, A. E. (2017). Studying the urgent updates of popular games on the Steam platform. *Empirical Software Engineering*, 22(4):2095–2126.

Lin, D., Bezemer, C.-P., and Hassan, A. E. (2018a). An empirical study of early access games on the Steam platform. *Empirical Software Engineering*, 23(2):771–799.

Lin, D., Bezemer, C.-P., Zou, Y., and Hassan, A. E. (2018b). An empirical study of game reviews on the Steam platform. *Empirical Software Engineering*, pages 1–38.

Linehan, C., Bellord, G., Kirman, B., Morford, Z. H., and Roche, B. (2014). Learning curves: analysing pace and challenge in four successful puzzle games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play*, pages 181–190. ACM.

Linstead, E. and Baldi, P. (2009). Mining the coherence of gnome bug reports with statistical topic models. In *Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories*, pages 99–102. IEEE.

Liu, X., Lu, X., Li, H., Xie, T., Mei, Q., Mei, H., and Feng, F. (2017a). Understanding diverse usage patterns from large-scale appstore-service profiles. *IEEE Transactions on Software Engineering*.

Liu, Y., Liu, L., Liu, H., Wang, X., and Yang, H. (2017b). Mining domain knowledge from app descriptions. *Journal of Systems and Software*, 133:126–144.

Logfeller (2014). Release Date? - Underrail General Discussions. http://steamcommunity.com/app/250520/discussions/0/35222218811239879/#c626329186828487604. (last visited: Oct 16, 2018).

Long, J. D., Feng, D., and Cliff, N. (2003). *Ordinal Analysis of Behavioral Data*. John Wiley & Sons, Inc.

Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 431–439.

Löwgren, J. and Stolterman, E. (2004). *Thoughtful interaction design: A design perspective on information technology*. MIT Press.

Maalej, W., Happel, H.-J., and Rashid, A. (2009). When users become collaborators: towards continuous and context-aware user input. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object Oriented Programming Systems Languages and Applications*, pages 981–990. ACM.

Maalej, W. and Nabil, H. (2015). Bug report, feature request, or simply praise? on automatically classifying app reviews. In *Proceedings of the 23rd International Requirements Engineering Conference (RE)*, pages 116–125. IEEE.

Machkovech, S. (2018). Nintendo: letting our fans review video games might not be a good idea. https://arstechnica.com/gaming/2018/02/nintendo-has-opened-the-doors-to-fans-game-reviews-on-nintendo-com/. (last visited: Oct 16, 2018).

Man, Y., Gao, C., Lyu, M. R., and Jiang, J. (2016). Experience report: Understanding cross-platform app issues from user reviews. In *Proceedings of the 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 138–149. IEEE.

Manes, S. (1984). Taking a gamble with word vision. *PC Mag*, page 211.

Mäntylä, M. V., Khomh, F., Adams, B., Engström, E., and Petersen, K. (2013). On rapid releases and software testing. In *Proceedings of the 29th International Conference on Software Maintenance (ICSM)*, pages 20–29. IEEE.

Martens, D. and Johann, T. (2017). On the emotion of users in app reviews. In *Proceedings of the 2nd International Workshop on Emotion Awareness in Software Engineering*, pages 8–14. IEEE Press.

Martin, W., Harman, M., Jia, Y., Sarro, F., and Zhang, Y. (2015). The app sampling problem for app store mining. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 123–133. IEEE Press.

Martin, W., Sarro, F., and Harman, M. (2016). Causal impact analysis for app releases in google play. In *Proceedings of the 24th International Symposium on Foundations of Software Engineering*, pages 435–446. ACM.

Martin, W., Sarro, F., Jia, Y., Zhang, Y., and Harman, M. (2017). A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering (TSE)*, PP(99):1–32.

McDougall, J. (2010). Community heroes: Notch, for Minecraft. http://www.pcgamer.com/community-heroes-notch-for-minecraft/. (last visited: Oct 16, 2018).

McGraw, G. and Hoglund, G. (2007). Online games and security. *IEEE Security & Privacy*, 5(5):76–79.

McIlroy, S., Ali, N., and Hassan, A. E. (2016a). Fresh apps: an empirical study of frequently-updated mobile apps in the Google Play store. *Empirical Software Engineering*, 21(3):1346–1370.

McIlroy, S., Ali, N., Khalid, H., and Hassan, A. E. (2016b). Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 21(3):1067–1106.

McIlroy, S., Shang, W., Ali, N., and Hassan, A. (2017). Is it worth responding to reviews? a case study of the top free apps in the Google Play store. *IEEE Software*, 34(3):64–71.

McIntosh, S., Kamei, Y., Adams, B., and Hassan, A. E. (2016). An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering*, 21(5):2146–2189.

Medierra (2015). When is this game coming out? - Grim Dawn General Discussions. http://steamcommunity.com/app/219990/discussions/0/530649887209076036/#c530649887215003016. (last visited: Oct 16, 2018).

Medler, B., John, M., and Lane, J. (2011). Data cracker: developing a visual game analytic tool for analyzing online gameplay. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2365–2374. ACM.

Microsoft (2003). Understanding patch and update management: Microsoft's software update strategy. https://msdn.microsoft.com/en-us/library/cc768045.aspx. (last visited: Oct 16, 2018).

Microsoft (2015). Xbox Game Store. http://marketplace.xbox.com. (last visited: Oct 16, 2018).

Miller, J. (1991). Short report: Reaction time analysis with outlier exclusion: Bias varies with sample size. *The Quarterly Journal of Experimental Psychology*, 43(4):907–912.

Minecraft (2016). Minecraft official website. https://minecraft.net/. (last visited: Oct 16, 2018).

Motoyama, M., McCoy, D., Levchenko, K., Savage, S., and Voelker, G. M. (2011). An analysis of underground forums. In *Proceedings of the 2011 SIGCOMM Conference on Internet Measurement Conference (IMC)*, pages 71–80. ACM.

Mujahid, S., Sierra, G., Abdalkareem, R., Shihab, E., and Shang, W. (2018). An empirical study of android wear user complaints. *Empirical Software Engineering*, pages 1–27.

Murphy-Hill, E., Zimmermann, T., and Nagappan, N. (2014). Cowboys, ankle sprains, and keepers of quality: how is video game development different from software development? In *Proceedings of the 36th International Conference on Software Engineering*, pages 1–11. ACM.

Nagappan, M., Zimmermann, T., and Bird, C. (2013). Diversity in software engineering research. In *Proceedings of the 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, pages 466–476, New York, NY, USA. ACM.

Nayebi, M., Adams, B., and Ruhe, G. (2016). Release practices for mobile apps – what do users and developers think? In *Proceedings of the 23rd International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 552–562. IEEE.

Nayebi, M., Cho, H., and Ruhe, G. (2018). App store mining is not enough for app improvement. *Empirical Software Engineering*, pages 1–31.

NLTK Project (2017). Natural Language Toolkit. https://www.nltk.org/. (last visited: Oct 16, 2018).

Noei, E., Syer, M. D., Zou, Y., Hassan, A. E., and Keivanloo, I. (2017). A study of the relation of mobile device attributes with the user-perceived quality of android apps. *Empirical Software Engineering*, 22(6):3088–3116.

Ollsson, T., Toll, D., Wingkvist, A., and Ericsson, M. (2015). Evolution and evaluation of the model-view-controller architecture in games. In *Proceedings of the 4th International Workshop on Games and Software Engineering (GAS)*, pages 8–14. IEEE.

O'reilly, T. (2007). What is web 2.0: Design patterns and business models for the next generation of software. *Communications & strategies*, (1):17.

Orland, K. (2011). Minecraft Draws Over $33 Million In Revenue From 1.8M Paying Customers. http://www.gamasutra.com/view/news/33961/Minecraft_Draws_Over_33_Million_In_Revenue_From_18M_Paying_Customers.php. (last visited: Oct 16, 2018).

Orland, K. (2014). Introducing Steam Gauge: Ars reveals Steams most popular games. http://arstechnica.com/gaming/2014/04/introducing-steam-gauge-ars-reveals-steams-most-popular-games/. (last visited: Oct 16, 2018).

Oxford (2016). Oxford Dictionary Definition of Crowdfunding. http://www.oxforddictionaries.com/us/definition/american_english/crowdfunding. (last visited: Oct 16, 2018).

Pagano, D. and Maalej, W. (2013). User feedback in the appstore: An empirical study. In *Proceedings of the 21st International requirements engineering conference (RE)*, pages 125–134. IEEE.

Palomba, F., Linares-Vasquez, M., Bavota, G., Oliveto, R., Di Penta, M., Poshyvanyk, D., and De Lucia, A. (2015). User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*, pages 291–300. IEEE.

Palomba, F., Linares-Vásquez, M., Bavota, G., Oliveto, R., Di Penta, M., Poshyvanyk, D., and De Lucia, A. (2018). Crowdsourcing user reviews to support the evolution of mobile apps. *Journal of Systems and Software*, 137:143–162.

Palomba, F., Salza, P., Ciurumelea, A., Panichella, S., Gall, H., Ferrucci, F., and De Lucia, A. (2017). Recommending and localizing change requests for mobile apps based on user reviews. In *Proceedings of the 39th International Conference on Software Engineering (ICSE)*, pages 106–117. IEEE Press.

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., and Gall, H. C. (2015). How can i improve my app? classifying user reviews for software maintenance and evolution. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*, pages 281–290. IEEE.

Pascarella, L., Palomba, F., Di Penta, M., and Bacchelli, A. (2018a). How is video game development different from software development in open source? In *Proceedings of the 15th International Conference on Mining Software Repositories (MSR)*. IEEE.

Pascarella, L., Palomba, F., Di Penta, M., and Bacchelli, A. (2018b). How is video game development different from software development in open source? In *Proceedings of the 15th International Conference on Mining Software Repositories*, MSR '18, pages 392–402. ACM.

PCGamer (2014). Tim Schafer explains Spacebase DF-9's unexpected v1.0 release. `http://www.pcgamer.com/tim-schafer-explains-spacebase-df-9s-v1-0-release/`. (last visited: Oct 16, 2018).

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

Petrova, E. and Gross, N. (2017). 4 reasons people watch gaming content on youtube. `https://www.thinkwithgoogle.com/consumer-insights/statistics-youtube-gaming-content/`. (last visited: Oct 16, 2018).

Plunkett, L. (2018). Big Study Of 10 Million Steam Reviews Is Absolutely Fascinating. `https://kotaku.com/deep-study-of-10-million-steam-reviews-is-absolutely-fa-1825908713`. (last visited: Oct 16, 2018).

Politowski, C., Fontoura, L., Petrillo, F., and Guéhéneuc, Y.-G. (2016). Are the old days gone?: A survey on actual software engineering processes in video game industry. In *Proceedings of the 5th International Workshop on Games and Software Engineering*, pages 22–28. ACM.

Poretski, L. and Arazy, O. (2017). Placing value on community co-creations: A study of a video game'modding'community. In *Proceedings of the Conference on Computer Supported Cooperative Work and Social Computing*, pages 480–491. ACM.

Reyno, E. M. and Carsí Cubel, J. Á. (2009). Automatic prototyping in model-driven game development. *Computers in Entertainment (CIE)*, 7(2):29.

Romano, J., Kromrey, J. D., Coraggio, J., Skowronek, J., and Devine, L. (2006). Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen's d indices the most appropriate choices. In *Proceedings of the Annual meeting of the Southern Association for Institutional Research*.

Rosenmai, P. (2013). Using the median absolute deviation to find outliers. http://eurekastatistics.com/using-the-median-absolute-deviation-to-find-outliers/. (last visited: Oct 16, 2018).

RUBAT (2013). Censorship update. http://steamcommunity.com/games/252490/announcements/detail/1478602529560858502. (last visited: Oct 16, 2018).

Ruiz, I. J. M., Nagappan, M., Adams, B., Berger, T., Dienst, S., and Hassan, A. E. (2016). Examining the rating system used in mobile-app stores. *IEEE Software*, 33(6):86–92.

Ruiz, I. J. M., Nagappan, M., Adams, B., and Hassan, A. E. (2012). Understanding reuse in the android market. In *Proceedings of the 20th International Conference on Program Comprehension (ICPC)*, pages 113–122. IEEE.

Scacchi, W. (2011). Modding as a basis for developing game systems. In *Proceedings of the 1st international workshop on Games and software engineering*, pages 5–8. ACM.

Scacchi, W. and Cooper, K. M. (2015). Research challenges at the intersection of computer games and software engineering. In *Proceedings of the Conference on Foundations of Digital Games*.

Scalabrino, S., Bavota, G., Russo, B., Oliveto, R., and Di Penta, M. (2017). Listening to the crowd for the release planning of mobile apps. *IEEE Transactions on Software Engineering*.

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572.

Seaman, C. B., Shull, F., Regardie, M., Elbert, D., Feldmann, R. L., Guo, Y., and Godfrey, S. (2008). Defect categorization: making use of a decade of widely varying historical data. In *Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 149–157. ACM.

Shafiei, N. and van Breugel, F. (2013). Towards model checking of computer games with java pathfinder. In *Proceedings of the 3rd International Workshop on Games and Software Engineering (GAS)*, pages 15–21. IEEE.

Shores, K. B., He, Y., Swanenburg, K. L., Kraut, R., and Riedl, J. (2014). The identification of deviance and its impact on retention in a multiplayer game. In *Proceedings of the 17th conference on Computer supported cooperative work & social computing*, pages 1356–1365. ACM.

Sifa, R., Bauckhage, C., and Drachen, A. (2014). The playtime principle: Large-scale cross-games interest modeling. In *Proceedings of the 2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8.

Sifa, R., Drachen, A., and Bauckhage, C. (2015). Large-scale cross-game player behavior analysis on steam. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference*, pages 198–204. AAAI Press.

Sixen (2010). How to write a good bug report. https://us.battle.net/forums/en/sc2/topic/188789092. (last visited: Oct 16, 2018).

Smith, G., Othenin-Girard, A., Whitehead, J., and Wardrip-Fruin, N. (2012). Pcg-based game design: creating endless web. In *Proceedings of the International Conference on the Foundations of Digital Games*, pages 188–195. ACM.

Souza, R., Chavez, C., and Bittencourt, R. A. (2015). Rapid releases and patch backouts: A software analytics approach. *IEEE Software*, 32(2):89–96.

Stern, C. (2012). what makes a game indie: a universal definition. http://sinisterdesign.net/what-makes-a-game-indie-a-universal-definition/. (last visited: Oct 16, 2018).

Stone, J. (2016). New 'Steam stealer' malware gives hackers access to 77k users' games, credit card numbers every month. http://www.ibtimes.com/new-steam-stealer-malware-gives-hackers-access-77k-users-games-credit-card-numbers-2337423/. (last visited: Oct 16, 2018).

Summerville, A., Guzdial, M., Mateas, M., and Riedl, M. O. (2016). Learning player tailored content from observation: Platformer level generation from video traces using lstms. In *Proceedings of the Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

SuperData (2016). Market Brief Year in Review 2016. https://web.archive.org/web/20170702202939/https://www.superdataresearch.com/market-data/market-brief-year-in-review/. (last visited: Oct 16, 2018).

SuperData (2018). Worldwide digital games market: May 2018. https://www.superdataresearch.com/us-digital-games-market/. (last visited: Oct 16, 2018).

Takahashi, D. (2016). PwC: Game industry to grow nearly 5% annually through 2020. https://venturebeat.com/2016/06/08/the-u-s-and-global-game-industries-will-grow-a-healthy-amount-by-2020-pwc-forecasts/. (last visited: Oct 16, 2018).

Takkunen, P. (2015). Moving out of Early Access. http://store.steampowered.com/news/?appids=316080&headlines=1. (last visited: Oct 16, 2018).

Tantithamthavorn, C. and Hassan, A. E. (2018). An experience report on defect modelling in practice: Pitfalls and challenges. In *Proceedings of the International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP18)*, pages 286–295. ACM.

Tantithamthavorn, C., McIntosh, S., Hassan, A. E., and Matsumoto, K. (2017). An empirical comparison of model validation techniques for defect prediction models. *IEEE Transactions on Software Engineering*, 43(1):1–18.

Tassey, G. (2002). *The economic impacts of inadequate infrastructure for software testing*. National Institute of Standards and Technology.

Taylor, T. L. (2006). Beyond management: Considering participatory design and governance in player culture. *First Monday*.

ThunderPeel2001 (2015). The facts about Spacebase DF-9. https://steamcommunity.com/app/246090/discussions/0/616198900637753626/. (last visited: Oct 16, 2018).

Tian, Y., Nagappan, M., Lo, D., and Hassan, A. E. (2015). What are the characteristics of high-rated apps? a case study on free android applications. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*, pages 301–310. IEEE.

TimOfLegend (2014). Tim Schafer answers common questions about v1.0. https://steamcommunity.com/app/246090/discussions/0/613936673464943075/. (last visited: Oct 16, 2018).

Ullrich, C., Borau, K., Luo, H., Tan, X., Shen, L., and Shen, R. (2008). Why web 2.0 is good for learning and for research: principles and prototypes. In *Proceedings of the 17th international conference on World Wide Web*, pages 705–714. ACM.

Valve (2016a). Introducing Early Access. http://store.steampowered.com/earlyaccessfaq/. (last visited: Oct 16, 2018).

Valve (2016b). Steam Store. http://store.steampowered.com/. (last visited: Oct 16, 2018).

Valve (2018). Spacebase DF-9 on Steam. http://store.steampowered.com/app/246090/. (last visited: Oct 16, 2018).

Valve (2018). Steam community. https://steamcommunity.com/. (last visited: Oct 16, 2018).

van Angeren, J., Alves, C., and Jansen, S. (2016). Can we ask you to collaborate? analyzing app developer relationships in commercial platform ecosystems. *Journal of Systems and Software*, 113:430–445.

Varvaressos, S., Lavoie, K., Gaboury, S., and Hallé, S. (2017). Automated bug finding in video games: A case study for runtime monitoring. *Computers in Entertainment (CIE)*, 15(1):1.

Vasa, R., Hoon, L., Mouzakis, K., and Noguchi, A. (2012). A preliminary analysis of mobile app user reviews. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, pages 241–244. ACM.

Villarroel, L., Bavota, G., Russo, B., Oliveto, R., and Di Penta, M. (2016). Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, pages 14–24. ACM.

Vu, P. M., Nguyen, T. T., Pham, H. V., and Nguyen, T. T. (2015a). Mining user opinions in mobile app reviews: A keyword-based approach (t). In *Proceedings of the 30th International Conference on Automated Software Engineering (ASE)*, pages 749–759. IEEE.

Vu, P. M., Pham, H. V., Nguyen, T. T., et al. (2015b). Tool support for analyzing mobile app reviews. In *Proceedings of the 30th International Conference on Automated Software Engineering (ASE)*, pages 789–794. IEEE.

Vu, P. M., Pham, H. V., Nguyen, T. T., et al. (2016). Phrase-based extraction of user opinions in mobile app reviews. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 726–731. ACM.

Waclawek, T. (2015). Ronin, the turn-based action-platformer, is now on Steam. http://store.steampowered.com/news/?appids=274230&headlines=1. (last visited: Oct 16, 2018).

Walker, P. (2014). Early Access popularity growing, but only 25% have released as a full game. http://www.gamesindustry.biz/articles/2014-11-13-early-access-popularity-growing-but-only-25-percent-have-released-as-a-full-game. (last visited: Oct 16, 2018).

Wallner, G. (2015). Sequential analysis of player behavior. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, pages 349–358. ACM.

Wallner, G. and Kriglstein, S. (2012). A spatiotemporal visualization approach for the analysis of gameplay data. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1115–1124. ACM.

Wallner, G. and Kriglstein, S. (2013). Visualization-based analysis of gameplay data–a review of literature. *Entertainment Computing*, 4(3):143–155.

Wang, H., Li, H., Li, L., Guo, Y., and Xu, G. (2018). Why are android apps removed from google play?: a large-scale empirical study. In *Proceedings of the 15th International Conference on Mining Software Repositories*, pages 231–242. ACM.

Washburn Jr, M., Sathiyanarayanan, P., Nagappan, M., Zimmermann, T., and Bird, C. (2016). “What went right and what went wrong”: An analysis of 155 postmortems from game development. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, pages 280–289. IEEE/ACM.

Wei, L., Liu, Y., and Cheung, S.-C. (2017). Oasis: prioritizing static analysis warnings for android apps based on app user reviews. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, pages 672–682. ACM.

Welch, C. (2013). Steam Early Access lets gamers buy and play titles still in development. http://www.theverge.com/2013/3/20/4128644/steam-early-access-buy-and-play-games-still-in-development. (last visited: Oct 16, 2018).

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.

Wood, A. (2018). Steam study finds users leave negative reviews more quickly than positive ones. https://www.pcgamer.com/steam-study-finds-users-leave-negative-reviews-more-quickly-than-positive-ones/. (last visited: Oct 16, 2018).

Wow Wiki (2009). Hotfix - Vanilla WoW Wiki - Wikia. http://vanilla-wow.wikia.com/wiki/Hotfix. (last visited: Oct 16, 2018).

Yin-Poole, W. (2014). Valve tightens Steam Early Access rules for developers. http://www.eurogamer.net/articles/2014-11-21-valve-tightens-steam-early-access-rules-for-developers. (last visited: Oct 16, 2018).

YouTube (2018). Auto-generated topic channels - youtube help. https://support.google.com/youtube/answer/2579942. (last visited: Oct 16, 2018).

Zimmermann, T., Phillips, B., Nagappan, N., and Harrison, C. (2012). Data-driven games user research. In *Proceedings of the CHI Workshop on Game User Research (CHI-GUR 2012)*.

Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schroter, A., and Weiss, C. (2010). What makes a good bug report? *IEEE Transactions on Software Engineering*, 36(5):618–643.

Zimmermann, T., Premraj, R., Sillito, J., and Breu, S. (2009). Improving bug tracking systems. In *Proceedings of the 31st International Conference on Software Engineering-Companion Volume*, pages 247–250. IEEE.

Zwillinger, D. and Kokoska, S. (1999). *CRC standard probability and statistics tables and formulae*. Crc Press.