Empir Software Eng (2017) 22:2095–2126
DOI 10.1007/s10664-016-9480-2

Studying the urgent updates of popular games on the Steam platform

Dayi Lin[1] · Cor-Paul Bezemer[1] · Ahmed E. Hassan[1]

Published online: 13 December 2016

Abstract The steadily increasing popularity of computer games has led to the rise of a multi-billion dollar industry. This increasing popularity is partly enabled by online digital distribution platforms for games, such as Steam. These platforms offer an insight into the development and test processes of game developers. In particular, we can extract the update cycle of a game and study what makes developers deviate from that cycle by releasing so-called urgent updates. 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 define urgent updates as 0-day updates (updates that are released on the same day), updates that are released faster than the regular cycle, or self-admitted hotfixes. We conduct an empirical study of the urgent updates of the 50 most popular games from Steam, the dominant digital game delivery 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 paper, 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

Communicated by: Emerson Murphy-Hill

✉ Dayi Lin
dayi.lin@cs.queensu.ca

Cor-Paul Bezemer
bezemer@cs.queensu.ca

Ahmed E. Hassan
ahmed@cs.queensu.ca

[1] Queen's University, Kingston, ON, Canada

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.

Keywords Update cycle · Update strategy · Urgent update · Computer games · Steam

1 Introduction

The steadily increasing popularity of computer games has led to the rise of a multi-billion dollar industry, reaching an estimated revenue of $91.5 billion in 2015 (Sinclair 2015). The scale of this industry is demonstrated by the number of players which reaches 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 changing distribution strategy 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)). Digital distribution allows game developers to easily distribute game updates and new content to the players through online gaming communities, such as the Steam Community (Valve 2016a). Games purchases on digital distribution platforms reached a revenue of $61 billion (SuperData 2015) in 2015.

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 a valuable insight into the update behavior of a game developer. 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 introduce changes that can easily create new problems (Bohn 2000), and hence such updates should be avoided by game developers.

In this paper, we perform an empirical study on urgent updates of the 50 most popular games on Steam (Valve 2016b), a popular digital game distribution 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 study the update frequency, update consistency and update strategy of the studied games in a preliminary study. Our preliminary study shows 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 examine 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 focuses 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. (2016) study 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.

Paper Organization The rest of this paper is organized as follows. Section 2 provides background information on the Steam platform, on update strategies and related work. Section 3 presents the methodology that we use in our empirical study. Section 4 presents our preliminary study. Section 5 presents the findings of our empirical study. Section 6 discusses the threats to the validity of our study. Finally, Section 7 concludes the paper.

2 Background

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

2.1 Steam Gaming Platform

Steam is a digital game platform, developed by Valve Software, that helps users with the installation, updates and management of their computer games. There are currently over 8,100 games available through Steam and the platform has over 142 million active users (Galyonkin 2016).

Steam acts as a digital game library, as it helps users track their games. For example, users can install the games, that they own, on multiple computers through Steam. Steam helps users to handle ownership logistics, such as storing the license keys that are needed to play a game. In addition, Steam manages the update process for those games, if necessary. Another advantage of Steam is that it provides a unified platform for users of different operating systems, such as Windows and Linux.

Users can buy and download Steam games from the Steam Store (Valve 2016b) or from third-party vendors. To play a Steam game, users must register the game on the Steam platform and install the Steam client. The game is then playable once the user logs into Steam using the client. The Steam client will verify ownership of the game and automatically install any available updates. It is mandatory to install the latest update in order to play a game through Steam. As a result, players are always using the latest version of a game, even if the last update was a buggy update. There is no option for undoing or skipping an update of a Steam game.

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 1 lists all available channels with a brief description of the content of each channel. Various third-party dashboards, such as SteamSpy (Galyonkin 2016), collect a plethora of aggregated information from the Steam Community about Steam games.

Table 1 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.

2.2 Update Strategies

In this paper, we classify 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. In this paper, 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, we discuss our classification of games based on their update strategy.

2.3 Related Work

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

2.3.1 Mining Digital Gaming Platforms

Mining data from digital gaming platforms is an area that has been gaining attention recently. Most research in this area is focused on Steam or the Steam community.

Chambers et al. (2005) analyzed two years of game traffic on several gaming platforms, including Steam. They demonstrate the difficulty of providing enough resources at launch time of a game and they show that gamers are extremely difficult to please.

Several empirical studies have examined the social network of the Steam Community. Blackburn et al. (2011) study cheaters in the Steam Community. They analyze more than 12 million player profiles of which 700,000 are flagged as cheater and show 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) analyze the evolution of the Steam Community social network and examine user groups in the Steam community. Sifa et al. (2015) studied cross-game behaviour of players in the Steam Community. They analyze how players that play multiple games on Steam divide their playtime and which games are played by them.

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.

Our work is the first, to the best of our knowledge, that uses a digital gaming platform, i.e., the Steam platform, to analyze urgent updates of games from a software engineering perspective.

2.3.2 Software Engineering and Games

Several studies have examined various software engineering aspects of game development.

Ampatzoglou and Stamelos (2010) examine how software engineering practices are used in game development. They show that game developers adjust traditional software engineering methods to make them fit for game development. Apostolos et al. propose the employment of more elaborate empirical methods, i.e., controlled experiments and case studies, in game development research. Murphy-Hill et al. (2014) perform a study with 14 interviewees and 364 survey respondents to elicite substantial differences between video game development and traditional software development practices. Murphy-Hill et al. find that game developers are hesitant to use automated testing because these tests limit the creativity of game designers, as designers must adhere to the limitations of automated testing.

Washburn Jr et al. (2016) study 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 Jr et al. extract a set of best practices and pitfalls for game development. They show that planning at the early stage of game development is important.

Most prior work of software engineering practices in a games context focuses on the differences between software engineering practices for traditional software and for games. While we focus on urgent updates in games, we find that prior work on update strategies does not necessarily hold for game development.

Lewis et al. (2010) present a taxonomy of 11 types of failures in video games by surveying game failure videos on YouTube. We compare Lewis et al.'s taxonomy with the reasons we identify for releasing urgent updates in Section 5.

2.3.3 *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) show 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) show that releasing software faster than a competitor can lead to financial benefit despite the high cost of hotfixes.

Kerzazi and Adams (2016) study 345 releases of a large e-commerce web application and identify 17 recurrent root causes of botched releases, classified into four major categories. Hassan et al. (2016) study 1,000 emergency updates of over 10,000 mobile apps in the Google Play Store. Hassan et al. identify 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. suggest 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.

2.3.4 *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. 2016). 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. (2016) show 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. show 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) conduct 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). Mäntylä et al. state 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. conclude that the rapid release strategy does not have a significant impact on the product quality. Souza et al. (2015) study 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. find 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) conduct 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. show 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. find 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 extend their work (Khomh et al. 2015) and suggest that one of the major challenges when switching to rapid releases is to automate 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.

3 Methodology

In this section, we introduce the methodology of our empirical study of urgent updates of popular games. We detail how we select our subject systems and extract the needed data to conduct our study. Figure 1 gives an overview of our methodology.

3.1 Selecting Subject Systems

We select the 50 most popular games on Steam on January 12, 2016. The list of top 50 games is provided by Steam Charts (Gray 2016), a website that ranks games by the number of players on that day. Table 2 shows details about the 50 games that we selected for our study.

3.2 Collecting Update Notes

We use 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 Section 2, developers do not necessarily need to announce all updates that they make. We use 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 find 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 extract 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 3 shows an example of an update note for the *Team Fortress 2* game.

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

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

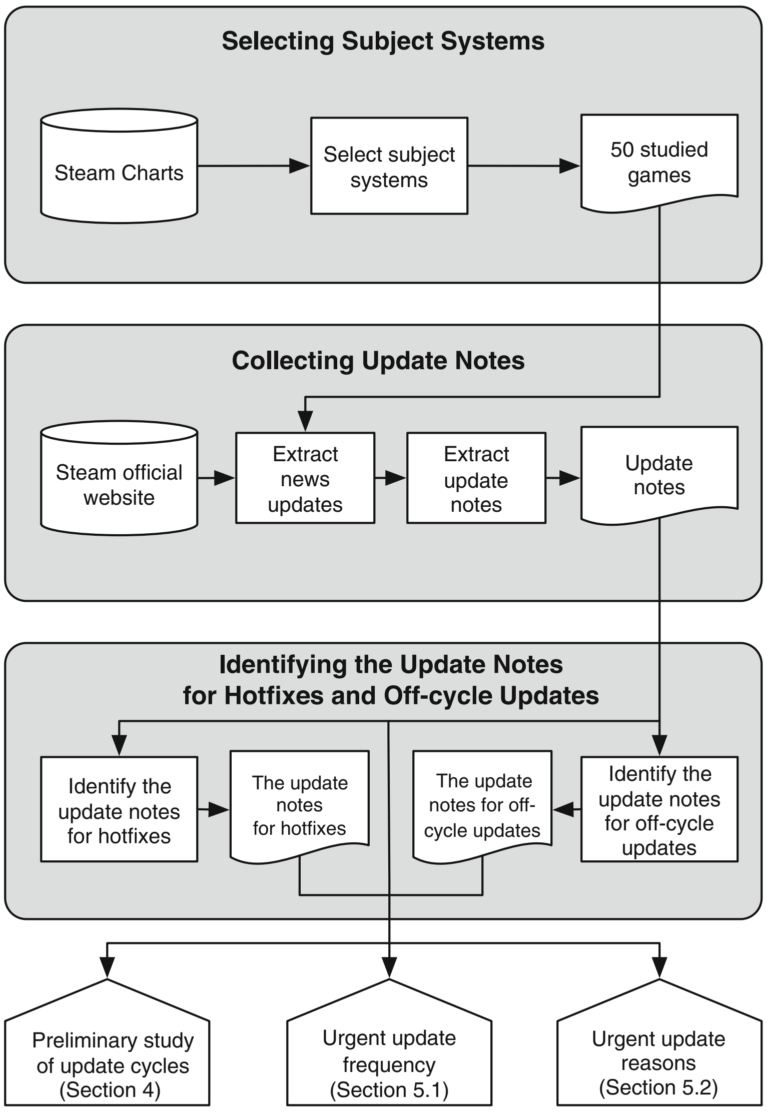

Selecting Subject Systems
Steam Charts
Select subject systems
50 studied games
Collecting Update Notes
Steam official website
Extract news updates
Extract update notes
Update notes
Identifying the Update Notes for Hotfixes and Off-cycle Updates
Identify the update notes for hotfixes
The update notes for hotfixes
The update notes for off-cycle updates
Identify the update notes for off-cycle updates
Preliminary study of update cycles (Section 4)
Urgent update frequency (Section 5.1)
Urgent update reasons (Section 5.2)

Fig. 1 Overview of our study

1. We keep all news updates that are posted on the *Product Release* or *Product Update* channel.
2. For the remainder of the news updates, we remove 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 remove 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.

Table 2 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	858,890	
Counter-Strike: Global Offensive	Valve	Action	2012	563,938	
Football Manager 2016	SPORTS INTERACTIVE	Sports	2015	68,949	
Fallout 4	Bethesda Game Studios	RPG	2015	61,214	
Grand Theft Auto V	Rockstar North	Adventure	2015	56,419	
Team Fortress 2	Valve	Action	2007	56,390	
ARK: Survival Evolved	Studio Wildcard	RPG	2015	50,522	✓
Sid Meier's Civilization V	Firaxis Games	Strategy	2010	45,352	
Garry's Mod	Facepunch Studios	Simulation	2006	39,694	
The Elder Scrolls V: Skyrim	Bethesda Game Studios	RPG	2011	36,107	
Warframe	Digital Extremes	Action	2013	35,983	
Rust	Facepunch Studios	RPG	2013	35,128	✓
Rocket League	Psyonix	Sports	2015	34,342	
Arma 3	Bohemia Interactive	Strategy	2013	32,294	
Counter-Strike	Valve	Action	2000	26,814	
H1Z1 : Just Survive	Daybreak Game Company	Adventure	2015	24,577	✓
Euro Truck Simulator 2	SCS Software	Simulation	2013	21,689	
Call of Duty: Black Ops III	Treyarch	Adventure	2015	21,643	
Terraria	Re-Logic	RPG	2011	20,594	
Unturned	Smartly Dressed Games	Casual	2014	20,466	✓
PAYDAY 2	OVERKILL - a Starbreeze Studio.	RPG	2013	17,064	
SMITE	Hi-Rez Studios	Action	2015	16,510	
The Witcher 3: Wild Hunt	CD PROJEKT RED	RPG	2015	14,415	
War Thunder	Gaijin Entertainment	Simulation	2013	14,364	
Path of Exile	Grinding Gear Games	RPG	2013	14,159	
Left 4 Dead 2	Valve	Action	2009	13,866	
Europa Universalis IV	Paradox Development Studio	Strategy	2013	13,112	
Counter-Strike: Source	Valve	Action	2004	13,068	
Tom Clancy's Rainbow Six Siege	Ubisoft Montreal	Action	2015	12,742	
DayZ	Bohemia Interactive	Action	2013	11,505	✓
Total War: ROME II - Emperor Edition[1]	Creative Assembly	Strategy	2013	10,795	
Trove	Trion Worlds	RPG	2015	10,216	
Mount & Blade: Warband	TaleWorlds Entertainment	RPG	2010	9,976	
Don't Starve Together	Klei Entertainment	Simulation	2014	9,876	✓

Table 2 (continued)

Title	Developer	Genre	Release year	# of players	Early access[2]
Borderlands 2	Gearbox Software	RPG	2012	9,720	
METAL GEAR SOLID V: THE PHANTOM PAIN [1]	Konami Digital Entertainment	Adventure	2015	9,513	
XCOM: Enemy Unknown	Firaxis Games	Strategy	2012	9,253	
Age of Empires II HD	Skybox Labs	Strategy	2013	8,523	
7 Days to Die	The Fun Pimps	Simulation	2013	8,253	✓
Cities: Skylines	Colossal Order Ltd.	Strategy	2015	7,369	
Company of Heroes 2	Relic Entertainment	Strategy	2013	7,074	
Arma 2: Operation Arrowhead	Bohemia Interactive	Strategy	2010	7,023	
AdVenture Capitalist	Hyper Hippo Games	Casual	2015	6,946	
Total War: ATTILA	Creative Assembly	Strategy	2015	6,843	
Hurtworld	Bankroll Studios	Simulation	2015	6,806	✓
Undertale	tobyfox	RPG	2015	6,748	
Brawlhalla	Blue Mammoth Games	Action	2015	6,530	✓
Just Cause 3	Avalanche Studios	Adventure	2015	6,518	
Dying Light: The Following - Enhanced Edition[1]	Techland	RPG	2015	6,360	
DARK SOULS II: Scholar of the First Sin[1]	FromSoftware, Inc	RPG	2015	6,342	

[1]We will use shortened game names throughout the rest of the paper 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

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 identify the following channels that are related to game reviews and the Steam client: *Rock. Paper. Shotgun*,

Table 3 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

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. Table 1 gives an overview of the channels that are relevant to our study.

We identify 2,672 update notes for the 50 studied games. In order to validate the precision and recall of our extraction steps, we manually analyze 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 check the identified update notes and remove 253 news updates that do not contain update notes, leaving 2,419 update notes for our study.

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

We distinguish two types of irregular updates in this paper:

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.

We identify 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 identify 163 update notes for self-admitted hotfixes. We manually check all 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 calculate the *days-between-updates* for all adjacent updates for all games. We then use 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 use 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 of MAD away from the median of the residuals, that value is classified as an outlier. However, Fig. 4 shows that the distributions of *days-between-updates* are highly unsymmetric. We address this problem by using the *Double MAD* as suggested by Rosenmai (2013), i.e., we calculate the MAD for the left and right side of the median of the distribution, and use 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

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

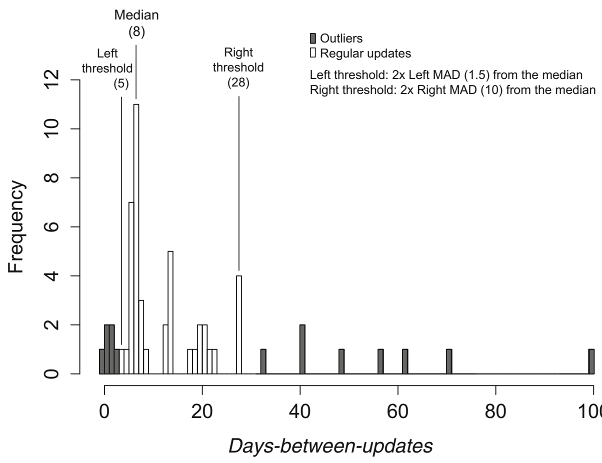


Fig. 2 An example of detecting outliers for the *Warframe* game

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 select 2 as the threshold for our dataset. Figure 2 shows an example of detecting outliers for the *Warframe* game using the *Double MAD*. We identified 411 off-cycle updates in total.

3.4 Dataset Description

Table 4 presents the description of our collected dataset.

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.

Table 4 Dataset description

# 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

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 use 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 remove games with less than 3 updates, as such games do not provide enough information to infer their update cycle. We calculate 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 calculate 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 5 shows the update frequency and update cycle consistency metrics for all studied games. We use these metrics to manually classify all the games into two classes: games that follow a frequent update strategy, and games that use a build-up candidate update strategy. For each studied game, we:

1. Examine the median and mode of the *days-between-updates*, and compare those numbers with the total number of updates.
2. Examine the update timeline of the game. Figure 3 shows the update timeline of the *War Thunder* game as an example.
3. Examine the update notes when necessary.
4. Classify 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, the first and the second author of this paper both did the classification independently, and then compared the results. Only 5 games were classified differently by the two authors, and the differences were easy to resolve after discussion. There was one game (the *War Thunder* game) which was classified into both classes. Figure 3 shows the update timeline of the *War Thunder* game. From Fig. 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 do not consider the data for the *War Thunder* game in the rest of our calculations to avoid confusion.

For each studied game, we calculate 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 use 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

Table 5 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

Table 5 Updates of studied games on Steam, sorted by the kurtosis of *days-between-updates* (as of January 12, 2016) (continued)

Title	Updates	% Self.adm hotfixes	% Off-cycle updates[2]	Days-between-updates[1]		
				Median	Mode[3]	Kurtosis
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

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 compute Cliff's delta d (Long et al. 2003) effect size to quantify the difference of the distributions. We use the following threshold for interpreting d, as proposed by Romano et al. (2006):

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

4.1 Update Frequency

Many studied games have periods in which they release frequently Table 5 shows that 20 out of 45 (44 %) of the studied games have a median *days-between-updates* that is

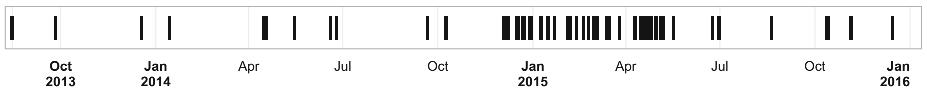


Fig. 3 Update timeline of the *War Thunder* game. Each *vertical line* represents an update

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 quick pace as well.

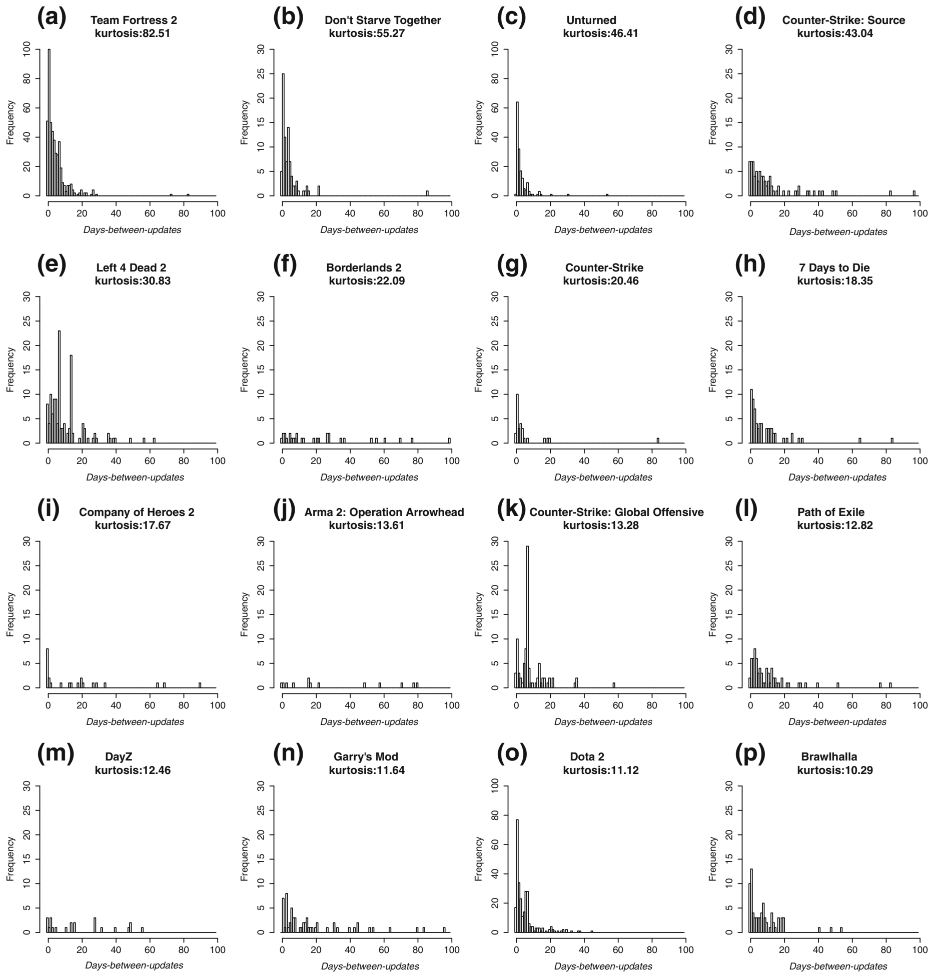


Fig. 4 Histogram of the *days-between-updates* of the 16 games with the highest kurtosis for that metric. *days-between-updates* greater than 100 are removed for clarity

4.2 Update Consistency

Most studied games do not have a consistent update cycle Figure 4 shows the distribution of the *days-between-updates* of the 16 games that have the highest kurtosis for the *days-between-updates* metric. Table 5 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 4f 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 4e 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 5 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.

4.3 Update Strategy

68 % of the studied games use a build-up candidate update strategy Table 6 shows the results of our update strategy classification. 68 % of the studied games follow the more traditional build-up candidate update strategy. 32 % percent 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 6 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 Civilization 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 5 shows the percentage of off-cycle updates of each studied game. Figure 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 % to 20 %,

Table 6 Update strategies and off-cycle updates of studied Steam games, sorted by the number of players (as of Jan 12, 2016)

Title	Update strategy		% Faster off-cycle updates[2]	% Slower off-cycle updates[2]	% 0-day updates[2]
	Frequent update	Build-up candidate			
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

Table 6 (continued)

Title	Update strategy		% Faster off-cycle updates[2]	% Slower off-cycle updates[2]	% 0-day updates[2]
	Frequent update	Build-up candidate			
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

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 6 shows the release timeline of the *Counter-Strike: Global Offensive* game. For clarity, we highlight the faster and slower off-cycle updates on separate timelines. We observe 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 6 shows the percentage of slower and faster off-cycle updates for each of the studied games. Figure 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.

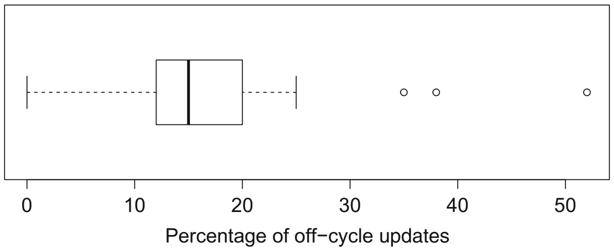


Fig. 5 Distribution of the percentage of off-cycle updates of studied games. Each data point represents a studied game

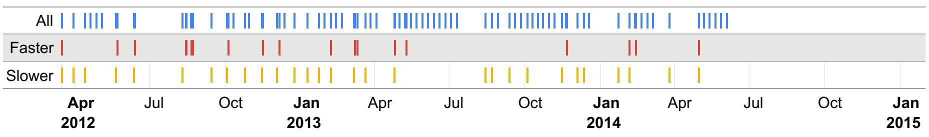


Fig. 6 Release timeline of the *Counter-Strike: Global Offensive* game. Each *vertical line* represents an update. There were no updates in 2015 and 2016, hence we omitted these years from the timeline for clarity

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 as slower off-cycle updates. We study the faster off-cycle updates in Section 5.1 and Section 5.2, and we discuss slower off-cycle updates in Section 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 8 shows the distribution of the percentage of faster off-cycle updates (of all updates). Figure 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.

5 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.

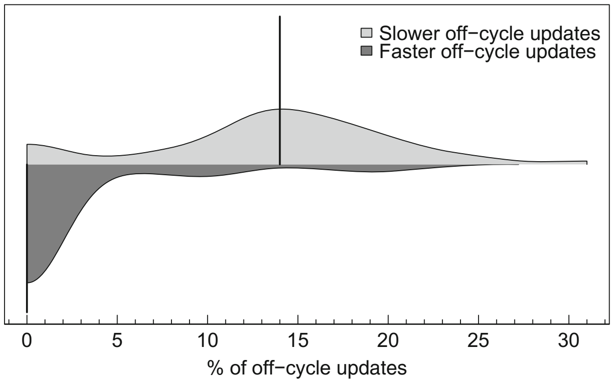


Fig. 7 Distribution of the percentage of off-cycle updates (of all updates) of each studied game. The *vertical lines* represent the median. The distributions are significantly different with a large effect size

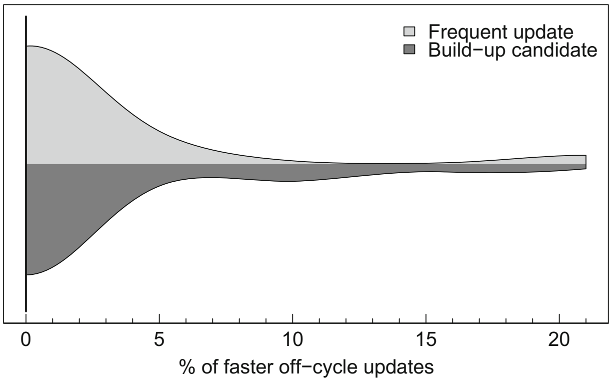


Fig. 8 Distribution of the percentage of faster off-cycle updates (of all updates) of each studied game. The *vertical lines* represent the median. The distributions are not significantly different (p > 0.05)

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 study the frequency of urgent updates. To study frequency, we analyze the data that we collected as described in Section 3. Second, we study the reasons that are given by developers in their update notes for releasing urgent updates. We manually extract and categorize the reasons for urgent updates from their update notes. We perform an

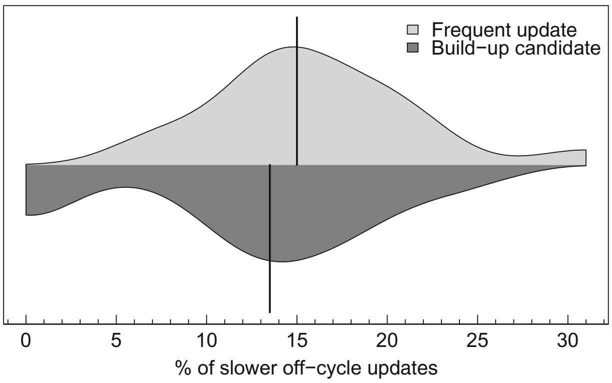


Fig. 9 Distribution of the percentage of slower off-cycle updates (of all updates) of each studied game. The *vertical lines* represent the median. The distributions are not significantly different (p > 0.05)

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 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 1: Our coding process for urgent updates

We manually examine the update notes for 162 0-day releases, 47 faster off-cycle updates, and 148 self-admitted hotfixes. We read all release notes and label 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 focus 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 identify 11 reasons from the update notes of urgent updates. Table 7 shows all reasons with their description and an example that is taken from a studied update note. The second author of the paper has manually validated the first author's analysis of reasons that are given in the update notes for urgent updates. The second author tagged a statistically-representative random 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 first author. Both authors 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 second author. Hence, after a short discussion, the disagreements were straightforward to resolve.

5.1 Urgent Update Frequency

80 % of the studied games have urgent updates 40 out of 50 (80 %) of the 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, 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

Table 7 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"*

strategies is significant, with a medium effect size. Figure 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

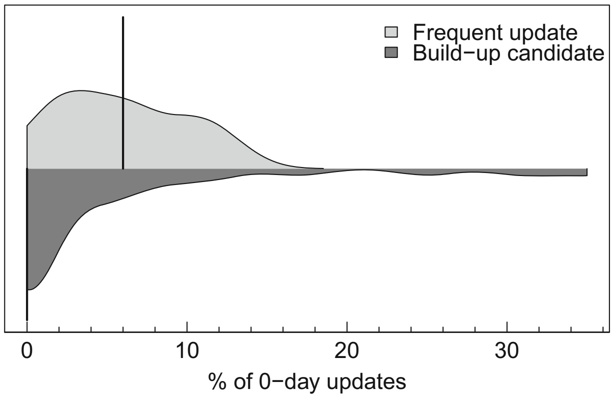


Fig. 10 Distribution of the percentage of 0-day updates (of all updates) of each studied game. The *vertical lines* represent the median. The distributions are significantly different with a medium effect size

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 5 shows that 23 out of 50 (46 %) of the 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 5 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 look into the early access games and find that the *Rust* game publishes its update notes on Twitter. We manually inspected the Twitter account of *Rust* and found 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 5 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 look 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 might be trying to hide 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).

5.2 Reasons for Releasing Urgent Updates

36 % of the urgent updates are released to make changes to the rules of a game Table 8 shows the frequency of each reason given in the update notes of urgent updates.

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

Table 8 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

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 calculate 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 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 8 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 9 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

Table 9 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

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 (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.

5.3 Comparison with Previous Work

As mentioned in Section 2.3, Lewis et al. identified 11 types of failures in video games by surveying game failure videos on YouTube. Table 10 shows a mapping of Lewis et al.'s taxonomy and the reasons that we found for releasing urgent updates. An interesting observation is that some of the reasons that we found 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.

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.

5.4 Discussion

As mentioned in Section 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 study the update timeline of all studied games and we observe that many games take longer to release an update as the age of the game increases. Figure 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,

Table 10 Mapping between Lewis et al.'s categories (Lewis et al. 2010) and the reasons for releasing urgent updates that are identified in this paper

This paper	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

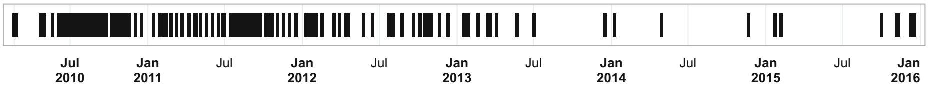


Fig. 11 Update timeline of the *Left 4 Dead 2* game. Each *vertical line* represents an update

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.

6 Threats to Validity

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

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 paper may be low bound estimates of the actual number of updates.

In our study, we assume 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.

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 Section 2, Steam is the largest digital distribution platform for PC gaming. Hence, popular Steam games are representative for a large number of games.

6.3 Construct Validity

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

We manually validated our approach for collecting update notes for self-admitted hot-fixes and found that our approach has a precision of 88 % and a recall of 87 %, as described in Section 3.2.

[4]Median Absolute Deviation, see Section 3.

7 Conclusion

In this paper, we study 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 conduct an empirical study on 2,419 update notes of the 50 most popular games on the Steam platform, a popular platform for digital game distribution. We use update notes to 1) identify the update strategy 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 try to hide 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 our paper is the finding that the choice of update strategy seems to affect the proportion of 0-day updates that developers have to release. We observe 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 show that releasing frequently leads to a higher proportion of patches that must be reverted.

Prior work (Khomh et al. 2012; Khomh et al. 2015; da Costa et al. 2016) on update strategies focuses 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 paper, we show that most games update much more frequently than once every six weeks, a phenomenon that was recently observed for mobile apps (McIlroy et al. 2016). 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.

References

Ampatzoglou A, Stamelos I (2010) Software engineering research for computer games: a systematic review. Inf Softw Technol 52(9):888–901

Arora A, Caulkins JP, Telang R (2006) Research note: sell first, fix later: Impact of patching on software quality. Manag Sci 52(3):465–471

Arora A, Krishnan R, Telang R, Yang Y (2010) An empirical analysis of software vendors' patch release behavior: impact of vulnerability disclosure. Inf Syst Res 21(1):115–132

Becker R, Chernihov Y, Shavitt Y, Zilberman N (2012) An analysis of the Steam community network evolution. In: Proceedings of the 27th convention of electrical & electronics engineers in Israel (IEEEI). IEEE, pp 1–5

Blackburn J, Simha R, Kourtellis N, Zuo X, Long C, Ripeanu M, Skvoretz J, Iamnitchi A (2011) Cheaters in the Steam community gaming social network. arXiv preprint arXiv:11124915

Bohn R (2000) Stop fighting fires. Harv Bus Rev 78(4):82–91

Chambers C, Feng WC, Sahu S, Saha D (2005) Measurement-based characterization of a collection of on-line games, USENIX Association

da Costa DA, McIntosh S, Kulesza U, Hassan AE (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 (MSR). ACM, pp 374–385

Galyonkin S (2016) SteamSpy - All the data and stats about Steam games. http://steamspy.com/, (last visited: Jul 16, 2016)

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

Hassan S, Shang W, Hassan AE (2016) An empirical study of emergency updates for top Android mobile apps. Empir Softw Eng:1–42

Huang J, Zimmermann T, Nagapan N, Harrison C, Phillips BC (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). ACM, pp 695–704

Kerzazi N, 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), vol 1. IEEE, pp 574–583

Khomh F, Dhaliwal T, Zou Y, Adams B (2012) Do faster releases improve software quality? an empirical case study of mozilla firefox. In: 2012 9th IEEE working conference on mining software repositories (MSR). IEEE, pp 179–188

Khomh F, Adams B, Dhaliwal T, Zou Y (2015) Understanding the impact of rapid releases on software quality. Empir Softw Eng 20(2):336–373

Kim BC, Chen PY, Mukhopadhyay T (2011) The effect of liability and patch release on software security: the monopoly case. Prod Oper Manag 20(4):603–617

Lesensmer (2013) Hotfix - League of Legends Wiki - wikia. http://leagueoflegends.wikia.com/wiki/Hotfix/, (last visited: Jul 16, 2016)

Lewis C, Whitehead J, 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). ACM, pp 108–115

Leys C, Ley C, Klein O, Bernard P, Licata L (2013) Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. J Exp Soc Psychol 49(4):764–766

Long JD, Feng D, Cliff N (2003) Ordinal analysis of behavioral data. John Wiley & Sons, Inc

Mäntylä MV, Khomh F, Adams B, Engström E, Petersen K (2013) On rapid releases and software testing. In: Proceedings of the 29th international conference on software maintenance (ICSM). IEEE, pp 20–29

McGraw G, Hoglund G (2007) Online games and security. IEEE Secur Priv 5(5):76–79

McIlroy S, Ali N, Hassan AE (2016) Fresh apps: an empirical study of frequently-updated mobile apps in the Google Play store. Empir Softw Eng 21(3):1346–1370

Microsoft (2003) Understanding patch and update management: Microsoft's software update strategy. https://msdn.microsoft.com/en-us/library/cc768045.aspx, (last visited: Jul 16,2016)

Microsoft (2015) Xbox Game Store. http://marketplace.xbox.com, (last visited: Jul 16, 2016)

Miller J (1991) Short report: Reaction time analysis with outlier exclusion: bias varies with sample size. Q J Exp Psychol 43(4):907–912

Motoyama M, McCoy D, Levchenko K, Savage S, Voelker GM (2011) An analysis of underground forums. In: Proceedings of the 2011 SIGCOMM conference on internet measurement conference (IMC). ACM, pp 71–80

Murphy-Hill E, Zimmermann T, 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. ACM, pp 1–11

Nayebi M, Adams B, 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). IEEE, pp 552–562

Romano J, Kromrey JD, Coraggio J, Skowronek J, 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: 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: Jul 16, 2016)

RUBAT (2013) Censorship update. http://steamcommunity.com/games/252490/announcements/detail/1478602529560858502, (last visited: Jul 16, 2016)

Seaman CB (1999) Qualitative methods in empirical studies of software engineering. IEEE Trans Softw Eng 25(4):557–572

Seaman CB, Shull F, Regardie M, Elbert D, Feldmann RL, Guo Y, Godfrey S (2008) Defect categorization: making use of a decade of widely varying historical data, ACM

Sifa R, Drachen A, Bauckhage C (2015) Large-scale cross-game player behavior analysis on Steam. In: Proceedings of the 11th artificial intelligence and interactive digital entertainment conference (AIIDE). AAAI

Sinclair B (2015) Gaming will hit $91.5 billion this year. http://www.gamesindustry.biz/articles/2015-04-22-gaming-will-hit-usd91-5-billion-this-year-newzoo, (last visited: Jul 16, 2016)

Souza R, Chavez C, Bittencourt RA (2015) Rapid releases and patch backouts: a software analytics approach. IEEE Softw 32(2):89–96

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: Jul 16, 2016)

SuperData (2015) Worldwide digital games market. https://www.superdataresearch.com/blog/us-digital-games-market/, (last visited: Jul 16, 2016)

Tassey G (2002) The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology

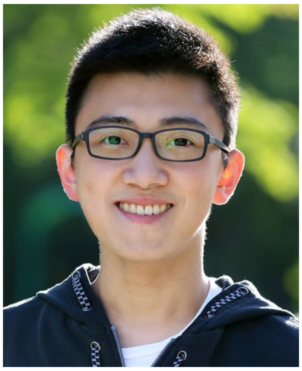
Valve (2016a) Steam community. http://steamcommunity.com/, (last visited: Jul 16, 2016)

Valve (2016b) Steam Store. http://store.steampowered.com/, (last visited: Jul 16, 2016)

Washburn Jr M, Sathiyanarayanan P, Nagappan M, Zimmermann T, 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). IEEE/ACM, pp 280–289

Wiki W (2009) Hotfix - Vanilla WoW Wiki - Wikia. http://vanilla-wow.wikia.com/wiki/Hotfix, (last visited: Jul 16, 2016)

Zwillinger D, Kokoska S (1999) CRC standard probability and statistics tables and formulae. Crc Press



Dayi Lin is a Ph.D. student in the Software Analysis and Intelligence Lab (SAIL) at Queen's University, Canada. His research interests include mining software repositories and empirical software engineering, in particular, Software Engineering related aspects on PC games. His research aims at providing a better understanding of Software Engineering aspects on PC games, to help game developers produce games with better quality and player satisfaction. Contact him at dayi.lin@cs.queensu.ca. More information at http://lindayi.me.

Cor-Paul Bezemer currently works as a postdoctoral research fellow in the Software Analysis and Intelligence Lab (SAIL) at Queen's University in Kingston, Canada. His research interests cover a wide variety of software engineering and performance engineering-related topics, including repository mining and performance regression analysis. His work has been published at premier software engineering venues such as the ESEC-FSE, ICSME, ICPE and SANER conferences. He was born in The Hague (Den Haag) in the Netherlands. Before moving to Canada, he studied at Delft University of Technology, where he received his BSc (2007), MSc (2009) and PhD (2014) degree in Computer Science. The title of his PhD thesis was "Performance Optimization of Multi-Tenant Software Systems".

Ahmed E. Hassan is the Canada Research Chair (CRC) in Software Analytics, and the NSERC/BlackBerry Software Engineering Chair at the School of Computing at Queen's University, Canada. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. Hassan received a PhD in Computer Science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community. Hassan also serves on the editorial boards of IEEE Transactions on Software Engineering, Springer Journal of Empirical Software Engineering, Springer Journal of Computing, and PeerJ Computer Science. Contact ahmed@cs.queensu.ca. More information at: http://sail.cs.queensu.ca/.