

A Survey of Video Game Testing

Cristiano Politowski
Concordia University
Montreal, QC, Canada
c_polito@encs.concordia.ca

Fabio Petrillo
Université du Québec à Chicoutimi
Chicoutimi, QC, Canada
fabio@petrillo.com

Yann-Gaël Guéhéneuc
Concordia University
Montreal, QC, Canada
yann-gael.gueheneuc@concordia.ca

Abstract—Video-game projects are notorious for having day-one bugs, no matter how big their budget or team size. The quality of a game is essential for its success. This quality could be assessed and ensured through testing. However, to the best of our knowledge, little is known about video-game testing. In this paper, we want to understand how game developers perform game testing. We investigate, through a survey, the academic and gray literature to identify and report on existing testing processes and how they could automate them. We found that game developers rely, almost exclusively, upon manual play-testing and the testers’ intrinsic knowledge. We conclude that current testing processes fall short because of their lack of automation, which seems to be the natural next step to improve the quality of games while maintaining costs. However, the current game-testing techniques may not generalize to different types of games.

I. INTRODUCTION

In the game industry, the first impression is of utmost importance; therefore, only high-quality games can expect to succeed. In December of 2020, the game “Cyberpunk 2077”¹ was released after seven years of development², hundreds of millions of dollars³, multiple delays⁴, and high expectations. Upon release, it immediately received strong criticism for its “buggy state” and, no long after, it was even removed from the PlayStation Store⁵ because it was not delivering the expected experience to its users, causing revenue loss and other collateral effects, like loss of prestige and good-will for the company CD Projekt Red⁶.

Many observers in industry and academia ponder how a company with a reputation for quality games⁷, like CD Projekt Red, with no shortage of money, time, skill, and experience could release a game with such poor quality. These observers, as well as personnel from the company itself, blame the scope, the management, and the *lack of testing*⁸ for the poor quality of the game. According to the company CEO, Marcin Iwiński, “Every change and improvement needed to be tested, and as it turned out, our testing did not show a big part of the issues you experienced while playing the game”⁹.

Thus, like with any piece of software, to deliver high-quality games, game developers must test their games rigorously during their development. In traditional software development, tests are considered essential (unit, component, integration, or end-to-end tests), and so is their automation [1]. Yet, to the best of our knowledge, there exists no comprehensive and systematic report on how game developers test (or not) their games. Thus, we ask the question: “What is the testing process in game development?”

To answer this question, we study the field of testing in game development by surveying the processes, techniques, gaps, concerns, and point-of-views using two different sources: (1) the *academic literature* (journal and conference papers, books, and theses) and (2) the *gray literature* (presentations in well-known conferences, blog posts from game developers, and video-game postmortems).

We report that manual game-play testing, a form of end-to-end testing also called play-testing, is the primary testing technique used by game developers. Automate play-testing sessions is difficult, which lead to developers hiring specialised testers to manually play-test their games, resulting in a game industry that relies almost exclusively upon manual labour and human expertise to test their games.

We conclude that the automation of video-game testing requires research and tools that game developers can integrate into their testing processes. Therefore, allowing them to create more systematic, repeatable, and reliable games, while game testers focus on player-centered aspects of the games, like game play or aesthetics.

The paper is structured as follows: Section II describes the method we used. Section III shows the results of our analysis of the academic literature. Section IV shows the results of our study of the gray literature. Section V discusses our results. Section VI describes threats to validity of our study. Section VII concludes the paper with future work.

II. SURVEY METHOD

Our approach is based on categories of reliability of the literature [2], with *white literature*, peer-reviewed papers, is the most reliable, followed by three gray literature tiers: first-tier, books, books chapters, government reports; *second tier*, annual reports, news articles, videos, presentations, Wiki articles; and, *third tier*, blogs, emails, tweets, letters.

We considered: (1) *academic literature*, journal, conference, and workshop papers, books and theses, and (2) *gray litera-*

¹<https://www.cyberpunk.net/ca/en/>

²<https://www.youtube.com/watch?v=cGmWwFpNIHg>

³<http://bit.ly/2M41IHG>

⁴<http://bit.ly/39M5mbV>

⁵<https://www.playstation.com/en-ca/cyberpunk-2077-refunds/>

⁶<https://en.cdprojektred.com/>

⁷The previous game released by the same company, “The Witcher 3”, was considered a great success: <http://bit.ly/3ipMHqs>.

⁸<https://bloom.bg/2LNtNhc>

⁹<http://bit.ly/39N8epd>

ture, presentations in well-known conferences, blog posts from game developers, and video-game postmortems. We did not observe any duplication or extension (journal papers extending explicitly conference papers with new material). We performed the search for academic and gray literature on 2019/06/21.

A. Academic Literature

We used two sources of academic documents: Scopus¹⁰ and Engineering Village¹¹. The query was: (test OR testing OR verification OR validation OR qa OR "quality assurance" OR debugging OR prototyping) AND (game OR video-game OR "video game" OR "digital game") AND NOT (gamification OR "serious games" OR education OR teaching) AND LIMIT-TO (LANGUAGE, "English"). After the inclusion and exclusion criteria, the query yields **327 papers**.

The *inclusion* and *exclusion* selection criteria were:

- Inclusion criteria:
 - Papers must be about video game development;
 - Papers must be about or discuss testing;
- Exclusion criteria:
 - Papers not written in English;
 - Papers about gamification;
 - Papers about serious games;
 - Papers about using games for educational purposes;
 - Papers about using games for medical purposes.

We did not set and use any quality criteria.

a) *Filtering: Title Reading*: We read the title of each of 327 papers and divided them into *game development* and *testing*. After that filtering, we obtained **71 papers**.

b) *Filtering: Abstract Reading*: We read the abstracts of the 71 papers and *rejected* or *accepted* them based on our criteria. We kept **38 papers**.

c) *Coding: Full Reading*: We read all 38 papers, taking notes of the relevant subjects and references according to the grounded theory technique and selected the most relevant papers for the snowballing, **8 papers**.

d) *Snowballing*: We performed a complete snowballing process, backwards and forward, on the 8 papers, re-checking the title and abstract of each paper again to verify the paper's relevance. The final dataset¹² included **96 papers**.

B. Gray Literature

For the gray literature, we used the same inclusion and exclusion criteria than the academic papers. We used three main sources of gray literature:

- Postmortems of video-game projects
 - We used a database of postmortems [3] and kept only the ones related to testing. We analyzed each one of them and extracted the root cause of the problems.
- Conferences on game development

- We searched for talks about testing in game-developer conferences. We read/watched presentations and summarized the points related to game testing.
- Media specialized in game development
 - We searched for articles about testing on Web sites and blogs about game development. We selected the most relevant and summarized the main points.

We also searched specialized forums. We searched Stack-Exchange¹³ for questions with tags "testing"¹⁴ and found 79 questions but without relevant answers, about hardware compatibility (11 questions), game-play balancing (10 questions), and network testing (8 questions).

Finally, we queried Google Search: test* AND game AND development -job -course -company and used the five first result pages. We found 100 Web sites (blogs, forums, companies, papers, and courses) but they were irrelevant and/or did not add new information.

III. RESULTS - ACADEMIC LITERATURE

A. Game testing and its particularities

The game industry has different characteristics compared to traditional software development. It is a junction between design, business and software engineering [4]. In traditional software development, the product's objective is to provide a service while the game's goal is the entertainment [5]. Traditional software products, called "productivity applications", are designed to accomplish one or more tasks as necessary elements of a non-entertainment endeavour (e.g. business endeavours) [6]. On the other hand, a game is more than software as it must enthrall the user with its mechanics, despite its foundation relying on a well-written software [7]. Finally, game software aims to provide an experience rather than say productivity, and these cause divergences on the adopted practices [8].

Game development consists of iteration of experimentation while embracing the changes as the key for a good game design is the constant experimentation of new features instead of stick with a set of requirements [9]. Since game development requirements are continually changing (volatile and mutable [10]), developers prefer to embrace the change, as said by the developer: "Why should we bother with strict decisions when we can make a prototype in two hours to test things out?" [11]. Therefore, game development emphasizes evaluating user experiences and using the feedback to drive design iterations [8], although some authors report a balance between planned activities and improvised ones [12].

Some games have a particular way of profiting and, therefore, different ways to manage their development life-cycle. For example, mobile games are not designed to be sold once, but to keep changing (balancing the gameplay) and evolving (adding new features) constantly depending on the audience needs and behaviour [13]. Regarding the quality, software quality can be determined by how well the product performs

¹⁰www.scopus.com

¹¹www.engineeringvillage.com

¹²The dataset is available at <https://github.com/game-dev-database/game-testing>.

¹³gamedev.stackexchange.com

¹⁴<https://gamedev.stackexchange.com/questions/tagged/testing>

its intended functions. For game software, this includes the quality of the player's experience plus how well the game features are implemented [14].

1) *Game testers*: Gameplay testers have the heavy-duty of finding bugs and any other abnormality in the game. Game testers should test game quality by verifying gameplay, logical consistency, observability, progressive thinking, reasoning ability, and exhaustively testing features, game strategy, and functionality [15]. Therefore, testers should understand the principles and the characteristics behind games and especially understand the game development context [10].

Despite their usefulness, testers are often treated as less important compared to the developers and game designers [5], [16]. Often, they work in non-desirable conditions [16], and the setup of game testing is cumbersome for the developer. Testing changes will take longer as the programmer needs to set up the proper conditions within the game world to exercise that code path [17]. Finding bugs is repetitive, time-consuming and challenging for human testers, which could cause testers to overlook defects [18].

2) *Game testing*: Testing games have some different aspects compared to testing traditional software [19]. Video games are not considered productivity software; they are tested less intensely on accuracy and more intensely on the overall experience [20]. This type of testing is referred to *gameplay testing*, which is the way developers assess the quality of the game [21] and it is preferred in detriment of other techniques [19], [22] as it gives the team the feedback in the form of "enjoyment heuristics" [23]. Accordingly, test management is more ad-hoc in game development, relying heavily on the usability tests with users, not on the documented test plans or predefined test cases [24].

The game development process involves iterations around experiments, and it is expected that the game's code becomes "throwable" [13], making the tests obsolete. This practice is not unusual as the game industry is famous for creating new code with each new title. Even when the "same" engine or core code is reused for a new game, that engine or code is often modified, repaired, and improved upon [14]. These iterations lead to changes in how the game was conceptualized, designed and coded [12]. Late changes are allowed and even expected [24] with playtesting inspiring new features [25].

Games have different types of requirements [6]. Traditional test planning methods cannot satisfy the fun (entertainment and excitement) requirements [13]. Also, measures related to fun, entertainment, gameplay and other user experience aspects could be too complex to plan and evaluate software testing [10]. The main difference between testing a game and regular software lies in the game requirements and the scope definitions of the project [10].

Testing sessions are based mostly on black-box testing with human testers [15]. Despite the differences, testing in games is mandatory, as stated by this developer: "If a tester comes to say that this does not work, there is not fun in it, you really cannot leave that in the game, you have to fix it" [11]. "

Finally, game developers find it difficult to reuse their code, have less pressure on evolving their systems' architecture, and are less able to overview the requirements of a project correctly. Furthermore, they report more difficulties in performing automated unit tests than non-game developers [26]. More particularities in game testing are described by Santos, Magalhaes, Capretz, *et al.* [10], like requirements analysis, test planning, test design and execution.

3) *Test automation*: There are significant differences between games and non-games regarding game developers' difficulties to write automated tests [26]. There is less automation in game testing, for different reasons shown in Table I.

TABLE I
DIFFICULTIES IN TEST AUTOMATION FOR GAMES.

Difficulty	Description
Coupling	It is hard to write the automation given the coupling between the user interface (UI) and game mechanics [13], [19], [25].
Scope	Trying to cover all "paths" of the game could restrict the game design [10], [19], sometimes seen as contrary to agility preventing the fast pace of changes [18], [25]. Developers believe that they cannot cover everything, due the large search space of possible game states [25], therefore the effort is not worth [18].
Randomness	The non-determinism on games (multi-threading, distributed computing, artificial intelligence, and randomness) make it hard to hard to assert the correct behavior [18], [19]. In this case is a "emergent software" as its randomness is a feature and players' surprise is desirable [9]. You need access to the randomness part of the game, otherwise you may never reproduce the bug properly [14].
Changes	It is difficult to keep the automation as the game design change too often, even core mechanics [10], [27], making its documentation becoming obsolete too fast [9]. Also, the source code is temporary and highly likely to change as development progresses [18].
Cost	A software engineering is more expensive compared to game-play testers [18], [19]. The cost of hiring new developers to write additional lines of code into a game to support automation (as well as writing scripts and other test utilities, interfaces, and so on) can be more than it would cost to pay human testers [14]. Also, manual re-test after code changes (new build) are expensive due to reproducibility issues [18].
Time	Programmers usually don't test their games. They usually don't have time to do so [14] or they believe that writing tests takes time away from the actual development process [18].
Fun-factor	Capturing the "fun" with an automated test is not possible [18] as game testing is human centric and human behavior is difficult to automate [10]. Also, it is hard to automate games where many events are happening simultaneously. Therefore, automation is used mainly in simple tests [10].
Code	Automated testing code may not be bug-free, nor be reusable from one game to another, or from one platform to another [18]. Game developers have more difficulties than other developers when reusing code [26].

Despite the difficulties in applying automating tests in game development, there are some benefits of doing it, for example, reproducibility, quality, fewer bugs, better test focus, game stability, and fewer human testers [18].

B. Game testing concerns

As the workload for gameplay testing is tiresome [16], the testers often got motion sickness by constant re-playing the games [19]. In this sense, automation can reduce the burden on both developers and human testers, however it also can introduce cost to game production, and it required programmers with a specialized focus on automated testing [18]. At the same time, the lack of test automation in game development “hurts” the bug fixing process as it becomes harder to reproduce the steps [19].

Game developers overlook the SE practices. On testing, this happens because low-level testing is neglected for the sake of gameplay testing [19]. Regarding automated game testing, few studios use test automation alleging causes like development time, staff size, or little knowledge about these tools [22]. Usability tests and frameworks are also used within game development, but they are tailored specifically for the game domain [28]. Even pair and test-first programming, two practices well established in traditional software development, were not largely used by game companies [29].

In game development, software working according to its functional requirements is not enough; the game must be appealing for the user [30]. This appeal involves balancing the gameplay, which is unique in game development and a challenge for the developers [6].

Creating video games is a complex task. The game industry is on the edge of technical complexity. On top of that, the field of game development is hard to assess considering the restrictions of proprietary code [9].

C. Game testing challenges

a) Add game testing sooner to development pipeline:

Game studios start testing for *User Experience* of games too late in the development life cycle, sometimes as late as beta, which means that most of the feedback obtained from the tests is unlikely to have an impact on the final game [31]. Key game concepts should also be testing before release so the reactions of the players can be verified [32]. Every aspect of a game should be tested during the development and production phases. The most important aspect of testing for game developers is to integrate testing as part of the production phase to improve efficiency. To ensure the delivery of quality games to the market, developers must consider different testing options during the production phase [15].

b) *Instrument the game with meaningful logs and create tools to visualize the events:* Combine playtesting with play analytics and advanced visualizations (e.g., using synthetic, procedurally generated game worlds to visualize game play data sets and temporal relationships) [30].

c) *Add automating testing suitable for game development domain:* The use of manual testing and the challenges with automated testing in game systems highlight the need for a new set of methodologies to ease the developers’ ability to identify malfunctions and to enable automatic testing activities for games. Studies should investigate new record–replay tools, automated test-data generators, etc. [26].

d) *Define design patterns for game development:* As the GoF patterns [33] are specific for OO systems, game development patterns are good practices specific for game genres [9]. Aside from game design patterns, which other patterns, more related to source code and more similar to traditional software, could be applied when building a game? How the quality of code in games influence the game’s functional requirements?

e) *Create an universal game design language:* The creation of a “universal game design language” could allow the game developers reuse/extend a feature from game to game [9]. Can developers interchangeably develop a game regardless of the tools (game engine)?

IV. RESULTS - GRAY LITERATURE

A. Postmortems of video-game projects

We used a database of postmortem problems provided by a previous work [34]. Among all the problems, only 4% (50 out of 1,035) are about testing. The list of test-related postmortems is available on-line¹⁵, whose problems Table II summarizes.

TABLE II
SUMMARY OF THE TESTING PROBLEMS FOUND IN THE POSTMORTEMS.

Problem	N
Insufficient testing	22
Process and testing plans issues	18
Specific project requirements	13
Feedback	7
Scope	6
Reproducibility	2
Logging	2
No in-house QA	2
Combinatorial explosion	1
Bug fixing	1

Insufficient testing is the most common problem quoted by game developers. Eight postmortems cite lack of test as a problem (PM#4, PM#19, PM#33, PM#35, PM#38, PM#43, PM#45, PM#49). Five mention that more testing was need in the early phases of the development (PM#5, PM#6, PM#21, PM#22, PM#39) while two others in later phases (PM#7, PM#11). Lack of unit testing (PM#17) and regression testing (PM#18) were also cited as well as beta testing (PM#39), most of it because of time constraints (PM#9, PM#24, PM#26), and play-testing in general (PM#34). Finally, developers also mentioned problems regarding testing-tools setup, like continuous integration, testing systems, and automation (PM#6, PM#28).

Lack of unit testing and wrong decisions for the testing plan are well exemplified by the following quote:

“In the end, we slipped to the side of not having as many unit tests as we would have liked. (...) By the time we noticed those problems and wanted to start adding tests, it was too late because some of that code relied on non-unit test friendly APIs like UIKit or Box2d. (...) We even shipped with a few off edge cases that we knew were buggy but we didn’t dare fix weeks before submission.” – Postmortem PM#17

¹⁵<https://github.com/game-dev-database/game-testing>

The benefits of automated tests are illustrated by:

“With automated tests QA could have focused on other tasks and we would faster know if a certain map or feature were broken when running the automated tests” while “Without automated testing, every bug fix had a greater potential of introducing hidden bugs into the game, and this cost us a lot of time when regression issues weren’t identified until long after they were introduced.” – Postmortems PM#6 and PM#28

Specific projects’ requirements were heavily stressed by game developers. Support for multiple platforms and its difficulties were the requirements mentioned the most (PM#5, PM#12, PM#18, PM#30, PM#31, PM#36). Testing multi-player and/or online games is difficult for developers (PM#2, PM#46, PM#47, PM#49, PM#50). Other issues related to outdated video drivers (PM#30) and stress test (PM#42).

Game developers also cited problems with test plans and processes. Game developers complained that QA can block a development team (PM#3, PM#40). The test strategies were considered problems in many games (PM#1, PM#2, PM#19, PM#29, PM#37, PM#45). Other problems were the overestimation/underestimation of the time and effort necessary to test (PM#2, PM#17, PM#27, PM#41, PM#44, PM#47, PM#49, PM#50). Estimation issues regarding the infrastructure, mostly hardware concerns, were also mentioned by game developers (PM#6, PM#8, PM#20).

Problems with poor feedback and testing sessions were mentioned for several game projects (PM#9, PM#11, PM#16, PM#22, PM#30, PM#39, PM#45), e.g.:

“Most beta testers are young people who have a lot of time on their hands; that’s great for finding bugs, but it can also be a problem because some of them lack perspective.” and “I mean the kind of testing that finds bugs, not suggests new features.” Postmortems PM#12 and # PM#46

Some game projects have too large a scope to be properly tested. Scope is an issue also cited by game developers (PM#6, PM#14, PM#23, PM#26, PM#38, PM#41)

Other problems related to the reproducibility of the bugs (PM#13, PM#25) and bug-fixing (PM#3), logging problems (PM#25, PM#30), lack of in-house QA team (PM#28, PM#40), and the complexity of game mechanics (PM#15).

The reproducibility problem is exemplified by:

“The arena bug was subtle, although the results were not, and was not easy to reproduce. We were dimly aware that the bug existed, but its frequency was unclear. Because of the difficulty in reproducing it, the scope of testing the rest of the game and the small size of the QA team, the bug was not properly identified.” – Postmortem PM#13

The problems with game scope, game mechanics, and insufficient testing are well exemplified by the following quote:

“[the game] had 100 items and five playable characters. 70 percent of the items in [the character]

stack, and all the item abilities will affect [him] in some way, so there were so many variables to keep track of that all the testing in the world couldn’t have prepared us for launch. (...) In order to fully test all the variables we had in place, it would have taken hundreds of testers several days of extensive play time to fully debug this [game] – there were bugs that actually took 100,000+ people four weeks to find due to how buried and rare some of them were.” – Postmortem PM#15

B. Conferences on game development

We searched for talks in conferences specialized in game development like GDC¹⁶ and Digital Dragons¹⁷. Table III shows the most relevant talks about game testing.

TABLE III
CONFERENCES ON GAME DEVELOPMENT.

	Title	Url
CASE-1	Smart Bots for Better Games: Reinforcement Learning in Production	https://bit.ly/2Hi5lyJ
CASE-2	Automated Testing and Profiling for ‘Call of Duty’	https://bit.ly/37hf3w1
CASE-3	Automated Testing and Instant Replays in Retro City Rampage	https://bit.ly/2vuuSIS
CASE-4	It’s Raining New Content: Successful Rapid Test Iterations	https://bit.ly/31KNk5I
CASE-5	Automated Testing of Gameplay Features in Sea of Thieves	https://bit.ly/2ONBFh8

1) [CASE-1] *Smart Bots for Better Games: Reinforcement Learning in Production*: Ubisoft¹⁸ showcased their use of Reinforcement Learning (RL) in production to test games with bots that can evolve while playing. RL has been applied/studied mainly in two areas of game development: improving AI behavior and creating test assistants. The former got attention when the OpenAI team trained bots on Dota 2 (OpenAI Five)¹⁹. The latter approach uses agents to test games, which has been used by Ubisoft in Triple-A games like Far Cry: New Dawn (2019), which has an open world, and Rainbow Six Siege (2015), which is multiplayer.

The author also described three techniques to train agents using RL: learning from pixels [35], [36], from states [37], [38], and from simulation [39].

2) [CASE-2] *Automated Testing and Profiling for Call of Duty*: Electronic Arts (EA) is a large company that develops and publishes games. EA has many studios working on different games. The talk was about the franchise Call of Duty, which receives a new iteration almost every year since 2003.

The team developed a tool called Compass, designed to keep track of builds/testing. The team workflow with Compass consisted of five different checks during the development of the game: continuous integration (CI) module, “all maps”

¹⁶<https://www.gdconf.com/>

¹⁷<http://digitaldragons.pl/>

¹⁸<https://ubisoft.com>

¹⁹<https://openai.com/projects/five/>

testing, nightly tests, and maintenance. The tool work with the help of game-play testing. For example, when testers find a place with a low frame-rate, the tool record this location, adds more characters and collects metrics automatically.

3) [CASE-3] *Automated Testing and Instant Replays in Retro City Rampage*: The author used automated testing techniques to test his indie game, with a small scope. His solution was to record the inputs, in log files, and have the engine re-play them. It allowed him to track down bugs using a simple diff tool on the output files. Advantages of this input-record approach are: automated QA, easy to deal with multiple platforms, and easy to narrow down game-play bugs.

The author also suggested that the game code should be deterministic; the same inputs should yield the same outputs. Otherwise, the input-record approach will deal with different outputs every time, defeating its purpose. Aside from building a deterministic game engine, other things to consider are: initializing all variables, creating own (controllable) random method, and splitting each instance that controls different modules of the game. Finally, especially for multi-platform games, floating-point computations may differ, which might yield different behaviors.

4) [CASE-4] *It's Raining New Content: Successful Rapid Test Iterations*: Riot is the company developing League of Legends, which had 11 million daily players and 5 million concurrent players at the time of the talk. Thus, one bug, even a small glitch, could affect millions. Thus, they created a process with which testing is carried carefully in different steps to detect/prevent bugs.

Their process consists in daily play-tests with continuous delivery that uses days instead of weeks. They also use automated tests for performance and integration testing, with ad-hoc scripts to test the game loop and front end.

5) [CASE-5] *Automated Testing of Gameplay Features in Sea of Thieves*: Rare is the developer of Sea of Thieves, using Unreal Engine. They used the engine features to add tests cases at different levels of abstraction. Rare has more than 23,000 tests and 100,000 asset audit checks. With this automated test process, they managed to keep a constant number of bugs and avoid working overtime. They divided the test into:

- Actor testing (70%): This type of tests uses the Actor class of Unreal Engine to have lighter tests compared to integration tests, which have heavy dependencies and do not scale well;
- Unit testing (23%): This is the most basic test unit;
- Map testing (5%): These tests load portions of the game and check a small action, like the interaction between some actors and objects;
- Screenshot testing (1%): This type of tests is used to check the rendering output;
- Performance testing (less than 1%): These tests monitor the frame-rate of the game and eventual bottlenecks.

C. Media specialized in game development

Gamasutra is a portal on everything related to video games and game development. We use Google Search to

search Gamasutra for anything related to game and testing since 2010: `test* AND game site:gamasutra.com after:2010`. From the results, we retrieved the first 50 articles, excluded non-relevant articles from their titles and their contents, and retained the four articles in Table IV.

TABLE IV
GAMASUTRA ARTICLES

#	Title	URL
MEDIA-1	Testing for Game Development	https://ubm.io/35jhuNI
MEDIA-2	Differences between Software Testing and Game Testing	https://ubm.io/37ygEOj
MEDIA-3	Unit testing in video games	https://ubm.io/2ZOtZPR
MEDIA-4	Automated Testing: Building A Flexible Game Solver	https://ubm.io/2SWY2U2

1) [MEDIA-1] *Testing for Game Development*: In this article, the author argues that video games, aside of being big and complex, are also software with emergent behavior²⁰, i.e., unpredictable. Moreover, the complex game logic and the unpredictability of the users make it difficult to guarantee that games will “react gracefully”.

Aside of these difficulties, testing games benefits developers, e.g., by generating knowledge about the games. Testing is important because a game with bugs is not fun to play. Yet, automated testing is hard to achieve and does not replace manual testing, but removes repetitive (regression) testing.

The author also states that TDD, a common way to develop traditional software, might not be suitable when prototyping games due to the constant changes in the game design. Finally, the addition of new features should take precedence over bug fixing, i.e., developers should not spend time and effort on an game that may not even be fun to play.

2) [MEDIA-2] *Differences between Software Testing and Game Testing*: The author discusses the differences between testing games and traditional software. He argues that black box testing and automation is the same for software and game. However, he lists some particularities of game testing:

- User-Experience (UX) testing is not the same as fun-factor testing;
- Balance testing needs a vast knowledge of the game design and the target audience;
- Testing the game world or some level can be automated in unique ways, such as having bots moving randomly through the game world to see if they get stuck;
- AI testing requires that the testers understand what triggers the behavior of different types of non-playable characters (NPC) and how these triggers can be confused by different parameters;
- Multiplayer/Network testing requires many players simultaneously in the game world, with computer-controlled opponents and different game servers.

²⁰“Behavior of a system that is not explicitly described by the behavior of the components of the system, and is therefore unexpected to a designer or observer” (<http://wiki.c2.com/?EmergentBehavior>).

3) [*MEDIA-3*] *Unit testing in video games*: In this article, the author discusses some concerns related to unit tests in game projects. He stated that he worked on Triple-A projects that lasted more than 3 year, with 20 developers, and without a single unit test. Most developers were highly specialized, working only on a tiny fraction of the whole project; for example, terrain rendering, network, UI, game-play, etc.

4) [*MEDIA-4*] *Automated Testing: Building A Flexible Game Solver*: This article is one more example of an ad-hoc solution to automate game testing. They used a script describing the goals of each game level and a brute-force algorithm to reach these goals²¹. Although this solution is well-suited for their game, there are problems with hard-coded test scripts, for examples:

- Linear testing: actions are always in the same order;
- Cross quests: a test script cannot take into account variations or optional side quests;
- Up to date: if developers “forget” to report modifications to the adventure, the scripts may fail;
- Dependent upon development completion: it is difficult to test until the complete adventure is implemented;
- Logic: a change to a quest in one chapter may impact another part of the game.

V. DISCUSSION

A. *Finding #1: The testing strategies should take into account the particularities of game projects wrt. traditional software*

Observations - Game development process has particularities: The majority of the researchers agree that game development has particularities compared to traditional software development, mostly because of the final product goal: software as a “productivity” product and video game as an “entertainment” product. These differences reflect on the way the process of developing a game is conducted, mainly with requirements and addition of features along the development.

Implications - New game testing methods are necessary: These particularities of game development should be taken in consideration when devising a testing strategy for a game project. Applying traditional testing strategies, e.g. high emphasis on unit testing using Test Driven Development (TDD), to a game project might not bring the expected benefits. Yet, new testing strategies should be the results of the convergence of well-known software testing strategies, like unit testing, continuous integration (CI), and TDD. Finally, these particularities do not justify using only manual testing.

B. *Finding #2: Game testers should work alongside software testers to complement one another skills*

Observations - Game testers are not software testers: Game testers are mostly game-play testers, responsible for searching bugs and assessing the game experience, i.e., the fun factor. Although they are usually not considered part of the development team and have poor working conditions, their feedback is fundamental to the success of games.

Implications - Game and software testers must work together as a team: In game development, testers are professionals with specific skill sets related to game testing, and they are not engineers in general. Therefore, they write, automate, or script source code-related tests. They deal with games as black-boxes. Thus, the test team should include engineers who can automate tests.

Testing in game development is mostly²² done by a QA team. This professionals assess games as black boxes, playing the role of users (players), stressing the games as intended for their users. Game developers deal with coding tasks, like game-engine development. Drawing a parallel with traditional software development, game developers should be the responsible for creating and executing tests. Thus, testing in game projects should include game developers with knowledge of the source code and game testers with knowledge of the games.

As with traditional software development, studios might ignore testing for reasons like scope, budget, and time. In these cases, coverage of the games might not be a good metric to follow. Still, game developers should pursue ways to automatize the testing process, but also provide ways to aid the game testers in their task.

C. *Finding #3: Automation in game development is overlooked, as it relies on manual human testers*

Observations - Automation in video-game testing is not common: Automation is often overlooked because of coupling, scope, randomness, cost, requirements, time, domain restrictions, non-reusable code. Yet, automation has many benefits: reproducibility, quality, bug reduction, better test focus, game stability.

Implications - Lack of automation demands more (from) human testers: In game development, the concern (and budget) mostly focuses on play-testing the game. Thanks deep-learning algorithms, there are initiatives to train agents and play test the games. It is still too early for proper adoption (i.e., using a standard solution), but it is indeed a step forward to mitigate the burden of the repetitive tasks that game testers have to do.

D. *Finding #4: Search for the “fun-factor” and “balancing” the game are mainly executed by game-play testers*

Observations - Game testing employs exclusive techniques: According to the academic papers, there are testing techniques exclusive for game development, mostly related to game-play testing and the search for fun or game balancing. However, these new techniques are poorly explained or too abstract, making them hard to implement and automate. Like regression testing or smoke testing, some techniques are known for their use in traditional software development. Still, in game development, the approaches are different, relying more on game testing.

²¹https://www.youtube.com/watch?v=yXXTJ_EK0IQ

²²Especially with independent developers, teams sometimes cannot afford a separated QA team and must themselves test the games.

Implications - Techniques performed only by gameplay testers: The search for the fun-factor is not the only thing that game testers do. They also use their skills and empirical knowledge to investigate the games intelligently. These techniques are hard to automate.

E. Finding #5: There is no one-size-fits-all testing process for game projects as the games differ greatly

Observations - Big studios use ad-hoc testing techniques: Big studios with resources can develop testing processes tailored for their needs, well suited for specific games. Small studios rely on creativity and customized techniques to test their games. The lack of well-defined strategies is evident, even when using off-the-shelves engines like Unreal, developers must devise their own testing strategies.

Implications - The use of ad-hoc techniques shows that there is no testing standard for game development: Big studios can build tailored systems that are well integrated into their development pipelines. As game studios struggle to reach the release date, commonalities among different game types should be investigated, so that working strategies could be reused/tailored for specific games.

F. Finding #6: The acknowledge importance of testing by game studios should open the door for open-source initiatives

Observations - Game studios are aware of testing importance: Although not common, there are advanced testing techniques adopted by big companies. All projects described in the gray literature (Section IV) implemented a different testing technique, so it can be better applied to its game type. This fact allows us to believe that developers are aware of the importance of discovering new techniques to test the game and not rely only on play-testing sessions. We observed that different studies apply ad-hoc game testing strategies, varying according to game genres (FPS, RPG, Sports, etc), game types (2D, 3D), revenue models (game-as-a-service), etc.

Implications - The testing importance shows that the game industry should invest in open-source initiatives to spread the knowledge and advance the field: The game industry should learn from traditional software initiatives and invest in sharing knowledge among game developers, so that the field can grow faster.

G. Finding #7: Issues like lack of plan and poor testing coverage call for game testing to be performed early

a) Observations - Lack of plan and testing coverage are the main problems: The most common issues gathered from video-game postmortems indicate that game developers do not plan ahead testing and, consequently, test coverage is low. Moreover, the specific requirements of the project are listed as the main cause of difficulties in adding proper testing during the development. Other problems are related to feedback from the testers and project's scope.

b) Implications - Testing and gameplay testing should not be delayed: The non-linearity and randomness of most of the games nowadays, makes it hard to cover essential cases of a game. The scope—the world, the variables, the randomness, the paths—are too much to cover. Developers have concerns regarding the lack of testing. Testers need the game artifact in a “playable” version to perform their assessments. However, given the tight deadlines, this playable build is only accessible late in development. Game developers should strive to release incremental builds to game testers.

VI. THREATS

a) Hard to find trusted information: The most challenging aspect of this study was finding trusted information about video-game development. We saw biased data even in academic papers. Moreover, video-game studios are secretive about their processes and practices. We tried to mitigate this challenge by using three sources of information: academic papers, technical blogs and talks, and postmortem data.

b) Academic studies diverge greatly: Studies about games and software engineering discuss mainly the differences between systems and games, while AI papers focus on machine-learning models for mastering the game. Few papers show an end-to-end solution for testing games.

Our choice of using only titles and abstract to classify a paper as relevant may have led us to reject (many) relevant ones. However, these papers should have reappeared during the snowballing process (from 71 to 38 to 96).

c) Absence of a common field: Academic studies are spread across computer science (new algorithms), software engineering (tools and processes), video-game design (game play, aesthetics), and machine learning (training agents). Thus, points of view and vocabularies vary greatly. We kept this study focused on software engineering, but intersections/overlaps and contradictions still exist.

VII. CONCLUSION

No matter how large are their budgets or team sizes, video-game projects are notorious for having day-one bugs. Some games can recover, like *No Man's Sky* that had problems with missing features at launch but recovered its reputation after many updates along the years²³, while others are forgotten, like *Anthem*, which also had bugs at launch but which developers preferred to rewrite it entirely²⁴, finally ending up canceled²⁵.

In this paper, we studied video game projects by investigating game testing in the academic and gray literature. We surveyed processes, techniques, gaps, concerns, and point-of-views to understand how game developers test their projects.

The main findings are as follows:

- 1) Testing strategies must consider the particularities of game projects; unlike traditional software (Section V-A);
- 2) Game testers should work along with software testers to complement each other skills (Section V-B);

²³<http://bit.ly/3isBBkn>.

²⁴<http://bit.ly/3ir9PF3>

²⁵<https://www.pcgamer.com/anthem-next-cancelled/>

- 3) Automation is overlooked; game testing currently relies mostly on human testers (Section V-C);
- 4) The search for the “fun-factor” and “game balance” is mainly executed by game-play testers (Section V-D);
- 5) There is no one-size-fits-all testing process for game projects as the game types differ greatly (Section V-E);
- 6) Game studios acknowledge the importance of testing, opportunity for open-source initiatives (Section V-F);
- 7) Issues like lack of plan and poor testing coverage call for game testing to be performed early (Section V-G);

We conclude that automating the video-game testing process is the natural next step. However, developers and researchers lack processes, frameworks, and tools to help them with test automation. It results in *ad-hoc* techniques that are hard to generalize on different game types. Yet, testing is the key to quality games, and automation makes game quality sustainable.

In future work, we want to expand this survey into a complete, systematic mapping of the academic and gray literature, with clear research questions, quality criteria, and including topic (in)dependent classifications, like publication trends. We will compare the contents and trends of the academic and gray literature, in particular separating processes and techniques. We will also focus on practical techniques to automate the game testing, allowing designers, programmers, and game testers to work together.

REFERENCES

- [1] D. Spinellis, “State-of-the-Art Software Testing,” *IEEE Software*, vol. 34, no. 5, 2017, ISSN: 1937-4194. DOI: 10.1109/MS.2017.3571564.
- [2] R. J. Adams, P. Smart, and A. S. Huff, “Shades of grey: Guidelines for working with the grey literature in systematic reviews for management and organizational studies,” *International Journal of Management Reviews*, vol. 19, no. 4, pp. 432–454, 2017.
- [3] C. Politowski, F. Petrillo, G. C. Ullmann, J. de Andrade Werly, and Y.-G. Guéhéneuc, *Dataset of video game development problems*, 2020. arXiv: 2001.00491 [cs.SE].
- [4] S. Hyrynsalmi, E. Klotins, M. Unterkalmsteiner, T. Gorschek, N. Tripathi, L. B. Pompermaier, and R. Prikladnicki, “What is a Minimum Viable (Video) Game?” In *17th IFIP WG 6.11 Conference on e-Business, e-Services, and e-Society, I3E 2018*, vol. 11195, Springer-Verlag GmbH, Oct. 2018, pp. 217–231, ISBN: 3030021300. DOI: 10.1007/978-3-030-02131-3.
- [5] J. Kasurinen, M. Palacin-Silva, and E. Vanhala, “What Concerns Game Developers? A Study on Game Development Processes, Sustainability and Metrics,” *International Workshop on Emerging Trends in Software Metrics, WETSoM*, pp. 15–21, 2017, ISSN: 2327-0969. DOI: 10.1109/WETSoM.2017.3.
- [6] D. Calleele, P. Dueck, K. Wnuk, and P. Hynninen, “Experience requirements in video games: Definition and testability,” *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings*, pp. 324–333, 2015. DOI: 10.1109/RE.2015.7320449.
- [7] C. M. Kanode and H. M. Haddad, “Software engineering challenges in game development,” in *2009 Sixth International Conference on Information Technology: New Generations*, IEEE, 2009, pp. 260–265, ISBN: 9780769535968. DOI: 10.1109/itng.2009.74.
- [8] A. Osborne O’Hagan, G. Coleman, and R. V. O’Connor, “Software Development Processes for Games: A Systematic Literature Review,” *Communications in Computer and Information Science*, vol. 425, no. June, pp. 182–193, 2014, ISSN: 1865-0929. DOI: 10.1007/978-3-662-43896-1_16.
- [9] C. Lewis and J. Whitehead, “The whats and the whys of games and software engineering,” in *Proceeding of the 1st international workshop on Games and software engineering - GAS 11*, ACM Press, 2011, pp. 1–4, ISBN: 9781450305785. DOI: 10.1145/1984674.1984676.
- [10] R. E. S. Santos, C. V. C. Magalhaes, L. F. Capretz, J. S. Correia-Neto, F. Q. B. da Silva, and A. Saher, “Computer games are serious business and so is their quality,” in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM 18*, ACM Press, 2018, pp. 1–10. DOI: 10.1145/3239235.3268923.
- [11] J. Kasurinen, R. Laine, and K. Smolander, “How applicable is ISO/IEC 29110 in game software development?” In *Product-Focused Software Process Improvement*, vol. 7983 LNCS, Springer Berlin Heidelberg, 2013, pp. 5–19, ISBN: 9783642392580. DOI: 10.1007/978-3-642-39259-7_4.
- [12] P. Stacey and J. Nandhakumar, “A temporal perspective of the computer game development process,” *Information Systems Journal*, vol. 19, no. 5, pp. 479–497, Sep. 2009, ISSN: 1350-1917. DOI: 10.1111/j.1365-2575.2007.00273.x.
- [13] M. Mozgovoy and E. Pyshkin, “A comprehensive approach to quality assurance in a mobile game project,” in *Proceedings of the 14th Central and Eastern European Software Engineering Conference Russia on ZZZ - CEE-SECR 18*, ACM Press, 2018, pp. 1–8, ISBN: 9781450361767. DOI: 10.1145/3290621.3290835.
- [14] C. P. Schultz, R. Bryant, and T. Langdell, *Game Testing All in One (Game Development Series)*. Course Technology PTR, 2005, p. 520, ISBN: 1592003737.
- [15] S. Aleem, L. F. Capretz, and F. Ahmed, “Critical Success Factors to Improve the Game Development Process from a Developer’s Perspective,” *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 925–950, 2016, ISSN: 1000-9000. DOI: 10.1007/s11390-016-1673-z.
- [16] J. Schreier, *Quality Assured: What It’s Really Like To Test Games For A Living*, <https://kotaku.com/quality-assured-what-it-s-really-like-to-play-games-fo-1720053842>, Accessed: 2019-07-18.
- [17] J. Blow, “Game Development Harder Than You Think,” *Queue*, vol. 1, no. 10, p. 28, Feb. 2004, ISSN: 1542-7730. DOI: 10.1145/971564.971590.
- [18] S. Hooper, “Automated Testing and Validation of Computer Graphics Implementations for Cross-platform Game Development,” M.S. thesis, Auckland University of Technology, 2017.
- [19] E. Murphy-Hill, T. Zimmermann, and N. Nagappan, “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 - ICSE 2014*, ACM Press, 2014, pp. 1–11, ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568226.
- [20] M. Shelley, “On the Feasibility of using Use Case Maps for the Prevention of Sequence Breaking in Video Games,” Ph.D. dissertation, Carleton University, 2013, ISBN: 9780494946374.
- [21] R. Ramadan and B. Hendradjaya, “Development of game testing method for measuring game quality,” in *2014 International Conference on Data and Software Engineering (ICODSE)*, IEEE, Nov. 2014, pp. 1–6, ISBN: 9781479979967. DOI: 10.1109/icodse.2014.7062694.
- [22] J. Musil, A. Schweda, D. Winkler, and S. Biffl, “A Survey on the State of the Practice in Video Game Software Development,” *Institute of Software Technology and Interactive Systems*, Tech. Rep. March, 2010, p. 13.
- [23] A. Ampatzoglou and I. Stamelos, “Software engineering research for computer games: A systematic review,” *Information and Software Technology*, vol. 52, no. 9, pp. 888–901, Sep. 2010, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2010.05.004.
- [24] J. Kasurinen and K. Smolander, “What do game developers test in their products?” *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM’14)*, pp. 1–10, 2014, ISSN: 1949-3770. DOI: 10.1145/2652524.2652525.
- [25] H.-r. Ruonala, “Agile Game Development: A Systematic Literature Review,” M.S. thesis, University of Helsinki, Faculty of Science, Department of Computer Science, 2016.
- [26] L. Pascarella, F. Palomba, M. Di Penta, and A. Bacchelli, “How is video game development different from software development in open source?” In *Proceedings - International Conference on Software Engineering*, Gothenburg, Sweden, 2018, pp. 392–402. DOI: 10.1145/3196398.3196418.

- [27] J. Kasurinen, J.-P. Strandén, and K. Smolander, "What do game developers expect from development and design tools?" In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering - EASE 13*, ACM Press, 2013, p. 36, ISBN: 9781450318488. DOI: [10.1145/2460999.2461004](https://doi.org/10.1145/2460999.2461004).
- [28] A. I. Wang and N. Nordmark, "Software Architectures and the Creative Processes in Game Development," *Entertainment Computing – ICEC 2014*, vol. 8770, pp. 272–285, 2014. DOI: [10.1007/978-3-662-45212-7](https://doi.org/10.1007/978-3-662-45212-7).
- [29] J. Koutonen and M. Leppänen, "How are agile methods and practices deployed in video game development? a survey into finnish game studios," in *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, 2013, pp. 135–149. DOI: [10.1007/978-3-642-38314-4_10](https://doi.org/10.1007/978-3-642-38314-4_10).
- [30] W. Scacchi and K. M. Cooper, "Research Challenges at the Intersection of Computer Games and Software Engineering," *Computer Games and Software ...*, no. Fdg, 2015.
- [31] V. Paakkanen, "User Experience of Game Development Environments: Can making games be as fun as playing them?" M.S. thesis, Aalto University School of Science, 2014.
- [32] M. Washburn, P. Sathiyarayanan, M. Nagappan, T. Zimmermann, and C. Bird, "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 Companion - ICSE 16*, ACM Press, 2016, ISBN: 9781450342056. DOI: [10.1145/2889160.2889253](https://doi.org/10.1145/2889160.2889253).
- [33] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Addison-Wesley Professional, 1994, ISBN: 0201633612.
- [34] C. Politowski, F. Petrillo, G. C. Ullmann, and Y.-G. Guéhéneuc, *Game industry problems: An extensive analysis of the gray literature*, 2021. eprint: [2009.02440](https://arxiv.org/abs/2009.02440).
- [35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015, ISSN: 00280836.
- [36] J. Harmer, L. Gisslén, H. Holst, J. Bergdahl, T. Olsson, K. Sjöö, and M. Nordin, "Imitation learning with concurrent actions in 3d games," *CoRR*, vol. abs/1803.05402, 2018.
- [37] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=H1Dy--0Z>.
- [38] Y. Zhao, I. Borovikov, A. Beirami, J. Rupert, C. Somers, J. Harder, F. de Mesentier Silva, J. Kolen, J. Pinto, R. Pourabolghasem, H. Chaput, J. Pestrak, M. Sardari, L. Lin, N. Aghdaie, and K. A. Zaman, "Winning isn't everything: Training human-like agents for playtesting and game AI," *CoRR*, vol. abs/1903.10545, 2019.
- [39] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *CoRR*, vol. abs/1706.02275, 2017.