Testing practices in indie game development from a software engineering perspective: an exploratory study

Bachelor of Science Thesis in Software Engineering and Management

Snezhina Racheva

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Testing practices in indie game development
from a software engineering perspective: an exploratory study

Supervisor: Francisco Gomes de Oliveira Neto
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Testing practices in indie game development from a software engineering perspective: an exploratory study

Snezhina Racheva
Supervised by: Francisco de Oliveira Neto
University of Gothenburg
Gothenburg, Sweden
gusracsn@student.gu.se

***Abstract*—The games industry produces software with a main focus of entertainment, as opposed to fulfilling a certain business need. As a result, technical aspects of the final product are prioritised only at a base minimum level, keeping them from getting in the way of the game's entertainment value. Product testing is thus separated into these two concerns, with *playtesting* covering mostly gameplay design and *quality assurance* testing assessing the technical performance. However, literature lacks empirical evidence regarding the distinction from both types of testing, particularly, from a software engineering perspective. Moreover, this limited understanding is a hindrance for effective testing activities, since practitioners may not be familiar with known benefits and drawbacks in different software testing practices. This paper collects data from various sources bridge this gap in empirical evidence to: i) clarify the outlook of the game development industry from a software engineering perspective, ii) note down any particular challenges which might be faced due to this difference between the fields, and iii) propose solution from a software engineering practical perspective. Our findings reveal some insights about QA testing and playtesting, such as: i) the similarities between paytesting and exploratory testing, and ii) effects of using test-first approaches to support QA testing. Moreover, we report on problem and a list of suggested solutions to further bridge game development and software engineering.**

I. Introduction

While software is a major element in computer games, game development, in practice, is largely different from software engineering [1]. Software most often exists to serve a purpose (such as assist in a business task or stream a source of entertainment), whereas games' main purpose is predominantly to entertain. This distinction causes the development cycles of games and other software (or "business software") to also be vastly different, and, consequently, for game development to put very little focus on software engineering practices at all.

Viewed this way, game development can be separated into two separate aspects: i) the gameplay design, which is what provides the "fun" when playing a game, and ii) the software development aspect, which regards the implementation of the gameplay design itself. Seeing as the game development industry largely focuses on ensuring its products bring the users entertainment [2], its main goal in testing is to assess, namely, the fun-factor and gameplay.

This fun-assessing type of testing is also referred to as *playtesting*, coming from the method used to perform it, i.e., playing through the game either partially or in full, and taking notes on possible changes needed. It also bears resemblance to the exploratory approach [3] to testing - playtesting is also assessed, planned and performed simultaneously.

On the other hand, purely software quality testing (e.g., unit and integration testing) is referred to as Quality Assurance (QA) testing and is performed completely separately. For example, if a company is developing a platform-based game (e.g., Super Mario[1]), playtesting would determine whether the pace or difficulty of the game is well-balanced [4], the character's jumps are manageable and the reward at the end is sufficient. Conversely, QA testing would ensure glitches in calculations are avoided and the software behaves reliably and predictably.

Similar to QA testing, playtesting can also be used to spot technical issues and faults as well. For instance, a miscalculation in the code of the physics (i.e., a fault) of a platform-based game causing the player's jump action to act unpredictably (i.e., a failure) would be visible during a playthrough, and thus, easily spotted. This overlap in the results of testing is another reason for playtesting to take priority over types of QA testing.

Consequently, prioritising gameplay over software quality in testing can further increase the gap between game development and software engineering. Particularly, when considering the high pressure and demanding deadlines of delivering new products, any entertainment-centered industry would most likely be inclined to sacrifice quality over the apparent fun-factor. In the case of game development, this has led to a lack of a systematic and uniform approach to QA testing.

Another relevant factor is the availability of resources that a company can allocate to both QA testing and playtesting. By dividing companies based on their size (or number of employees), we can review big, well-established companies (e.g.,

[1]https://en.wikipedia.org/wiki/Super_Mario

Eletronic Arts[2] and Nintendo[3]), and small independent (indie) teams separately (e.g., Arrowhead Game Studios AB[4], Moon Studios GmbH[5]), as their budgets also differ significantly. As noted in "Playtesting for indie studios"[5],

> In the interest of minimising expenses, these [big-budget] teams budget for playtesting and quality assurance (QA). (...) Smaller game studios often do not have the budget for arranging expensive playtesting, so there is a need to incorporate playtesting techniques for smaller teams and accommodate their budgets.

For instance, in Electronic Arts, a well-established game development studio, practitioners highlight the usage of machine learning algorithms to automate the play-through of the games being developed [6], as well as manual qualitative assessment of game performance on a frames-per-second (FPS) level. Conversely, Indie studios' budgets commonly depend, not on their own company funds, but instead are often sourced to "game incubators" or otherwise external investors [5], and, as such, the budget allocated for playtesting and QA testing is smaller. The need for quality assurance in order to ensure an enjoyable final product is, however, just the same.

However, if one were to turn back towards the field of software testing for a solution to this issue, it might seem possible to minimise the time spent on designing test cases, and thus to decrease technical debt, by making test case design a vital part of the development process, namely through test-first development[7]. Similarly, the problems found when performing playtesting are often associated with exploratory-type testing (such as the difficulty in assessing test quality and coverage). Consequently, we argue that game development teams can leverage from existing solutions related to these challenges to achieve effective testing both in QA and playtesting.

The current state of research regarding game development as a software engineering discipline (and thus focusing on software engineering-related problems within it) is lacking - it is, specifically, in need of more empirical studies[1]. This thesis aims to assist in filling that gap in knowledge.

Specifically, our problem statement is that *verification and validation (V & V) activities in game development in indie companies often do not apply known advantages from established techniques in software testing state-of-the-art and practice*. Consequently, quality in game development is hindered. Our hypothesis is that using simple techniques, such as test-first [7] and exploratory testing [3], to leverage, respectively, QA testing and playtesting can help practitioners to achieve better testing cycles and manage technical debt.

If seen as ultimately leading to less technical debt (and encouraged QA testing), the test-first approach in indie game development settings has the potential to decrease the rift in quality between small and big-budget game developers, and to bring more software engineering practices (and thus - benefits) into game development as a whole.

At the same time we will be providing the research field with another empirical study into testing within game development, which, according to [1], there is a need for more of. Additionally, we will be further elaborating on game testing practices and their software engineering equivalents - something, which there is also a significant lack of research in.

A. Purpose of the study

The research aim of this paper is to explore possible ways for indie game development teams to improve their testing practises, by employing concepts from software engineering. Under the risk that QA testing is often neglected in such small-scope development environments, practitioners can benefit by finding a way to decrease the time needed for test case design (or even to introduce automated QA testing) in a non-disruptive manner, for example, by employing a test-first development methodology.

Furthermore, viewing playtesting from the perspective of software engineering (as opposed to gameplay design), can assist in smoothing out common issues and weaknesses within the assessment method. For instance, the bias of a player testing the game, or allowing playtesters to be creative when testing the games, are issues often seen in exploratory testing done in the gaming industry. Regarding such issues, the software testing research field might already be looking for a solution[3].

We will perform a case study to capture the current practices and tools used by practitioners (for both QA testing and playtesting), as well as identifying challenges, and proposing how these challenges can be addressed by state-of-the-art research in software testing. Moreover, future studies can focus on transferring these techniques to game development teams.

Particularly, the purpose of this study is to: i) empirically investigate how games are tested in production when developed by a small (max 10 people), independent team, ii) investigate how a test-first approach would potentially affect the testing process and the final quality of a game product, and iii) understand to what extent exploratory testing guidelines can aid playtesting.

B. Research Questions and/or Hypotheses

Considering that indie games companies understand the distinction between QA testing and playtesting (as indicated by literature—Section II), our hypothesis is that indie game companies can attain the benefits pertaining both activities by adopting software engineering practices. Particularly, we focus on two software testing techniques: test-first approaches and exploratory testing, given their commonalities to, respectively, QA testing and playtesting. Moreover, we aim to answer the following research questions:

[2]https://www.ea.com/
[3]https://www.nintendo.com/
[4]http://arrowheadgamestudios.com/
[5]https://www.orithegame.com/

RQ1: How is indie games' quality assessed throughout their production?
RQ2: To what extent can test-first approaches leverage QA testing in game development?
RQ3: How does playtesting relate to exploratory testing in terms of challenges met?

C. Summary of Contributions

Our exploratory study comprises data collection at an indie game company and analysis of artifacts from grey literature. These two data collection methods were analysed and the following contributions can be highlighted:

- We have clarified the separation between the different types of testing - playtesting and QA testing - as well as the expected results of both.
- We have compared playtesting to the very similar exploratory testing, and searched for any challenges they might have in common.
- We have made note of the effects TDD and TDD-like approaches have on QA practices in game development teams, as well as questioned the time and resource investment payoff.
- We have classified different approaches indie developers might have in order to tackle QA testing - outsourcing, as well as small-scale automated unit testing.

II. Review of the literature

When reviewing related literature to the problem domain, we focus on four separate topics: i) the distinction between game development and software engineering, ii) the distinction between playtesting and QA testing, and the overlap between the two, iii) test-first development in a software engineering setting, and its effects on development quality, and iv) exploratory testing in a software engineering setting, and the challenges and profit it brings.

In a systematic review, Ampatzoglou and Stamelos describe the software engineering-centered research related to games and game development [1] . The authors report on the lack of empirical research, specifically, controlled experiments and case studies, within the area of game development, and the need for more to be conducted. This research aims to fill that gap by performing a case study to collect data on the practices of testing in game development.

"Identifying usability and fun problems in a computer game during first use and after some practice"[2] explores usability- and fun-related fault detection during a first play-test of a game vs. when testing after getting accustomed to the gameplay. Its conclusion reports that the problems identified when first play-testing largely differ from the ones noted at a later time. This paper focuses mainly on usability and entertainment, which makes it largely irrelevant when it comes to QA testing research - something this thesis aims to focus on as well.

"Game Design Workshop(...)"[4], walks the readers through the authors' own definition of playtesting and its role within a game development environment, followed by detailed instructions on how to recruit and instruct playtesters, and collect and evaluate playtesting data. It's valuable that there is a clear distinction made between playtesting and QA testing here - even if it is sometimes contradicted by other sources. This book serves as a good source of context for someone looking into the game development industry and its practices, but it largely focuses on the gameplay design aspects, and less on software development (and furthermore, software engineering).

"Design for Research Results: Experimental Prototyping and Play Testing"[8] focuses on game design in a research context, and provides navigation through different types of prototyping and examples of playtesting. Its definition of QA testing as just another type of playtesting contradicts other sources, but also provides context on how vaguely-defined the terms really are. The article greatly differs from what this thesis aims to do, but provides assistance in understanding how to treat games in a research environment.

"Does Test-Driven Development Really Improve Software Design Quality?"[9] reviews whether the positive effect of test-driven development (or test-first approaches) is sufficient reason for their use. Our study will be looking at the positives brought by a change towards test-first development as well, so this paper provides additional insight into the methodology. At the same time, it does not involve itself with the game development field.

Regarding both types of testing we are investigating, [3] compares exploratory to scripted testing, highlighting their differences, strengths and weaknesses. While we can use this to compare playtesting to exploratory testing in software engineering, this also brings up the comparison between scripted and automated QA testing.

On the QA testing through test-first development side, "Specification by example"[7] theorises about technical debt being minimised as a result of test-first development. This is one of the possible outcomes to test-first game development we will be on the look out for.

III. Research Methodology

In order to answer the research questions, we performed an exploratory study (Table I). Through empirically studying the current testing practices in an indie game development company (or team), we aimed to compile a satisfactory source of data and reference for identifying challenges, and to propose improvements in software testing.

The subject of this case study is a game development team, unaffiliated with a large game development company (i.e. independent, or indie), which had, at the time of the interview, already established itself in the industry. The team had, as of April 16th, published two games across several platforms, and was, at the time, working on their third game. In addition to their experience in the industry, they are also alumni of one of the best-known game development university programmes in Sweden, and thus could testify for the prioritisation of testing and software quality in the programme's syllabus. In this paper they will be referred to as GameDevCo, in order to protect their anonymity.

Due to the limited research on common testing practices among indie development companies, the main goal of this interview was to establish the field under research: to clarify the current testing practices employed by the interviewed company and to generate suggestions for possible outcomes of the final case study.

In order to collect data for this base level of understanding, the interview questions needed to include a variety of aspects regarding testing in game development (e.g., unit testing, test frameworks, debugging of failures). Once the transcripts of the interviews were collected, we followed guidelines from qualitative in software engineering to analyze the data. This aimed to provide us with insight into testing-related problems the subjects of our study face, which we would then propose solutions to, using established software engineering approaches.

Once the proposal was prepared, we met a team representative once more, to introduce them to the possible solutions and receive further feedback. The thematic analysis of the first interview provided a base for the follow-up interview, where we were able to target the specific points of interest (as opposed to asking general questions), and were able to propose relevant solutions to problems GameDevCo is faced with.

The interview instrument is present in Appendix I.

The interview is spread out between the different plots of interest our research effort has - i) it begins with general assessment of the approach to testing at the company (RQ1), ii) it inquires about playtesting and the methods used to conduct it(RQ1, RQ3), iii) it covers QA testing practices in a question set, which is comparable to that of playtesting (RQ1, RQ2), and iv) it allows for developers to point out any additional testing practices we might have not mentioned before (RQ1).

This resulted in a clear picture of the developer team's approach to testing, which we then used in order to formulate relevant themes and areas in need of improvement.

After the first iteration of this paper was wrapped up, a second interview with GameDevCo was organized, where our findings were presented and discussed. This provided further insight into some areas of interest, as GameDevCo had made changes to their workflow, which coincided with changes proposed by the thesis, and could thus comment on the efficiency of the proposal.

However, a case study of this scope is insufficient to draw conclusions upon. As a way to provide further context into the area of testing in the game development field, we analyzed relevant posts from ”gray literature” sources, such as Gamasutra[6]. To provide even further (albeit non-scientific and purely anecdotal) context, we looked into game developer communities on Reddit[7]. As current scientific research within our subject field is largely limited, such sources could be useful

[6]http://www.gamasutra.com/

[7]https://www.reddit.com/ (ex. https://www.reddit.com/r/gamedev/

TABLE I
OUR EXPLORATORY STUDY PLANNING. WE FOLLOWED THE EMPIRICAL GUIDELINES SUGGESTED BY [10].

Objective	Explore
The context	QA testing in indie game development
The cases	An indie game development team
Theory	Software testing techniques
	Test-first approaches
	Exploratory testing
Research questions	RQ1, RQ2, RQ3 and RQ4
Methods	Qualitative data analysis
Selection strategy	Participants are game developers working with the Gothenburg Game Incubator
Unit of Analysis 1:	Focus on QA testing
	Test coverage, effort and costs.
Unit of Analysis 2:	Focus on playtesting
	Number of challenges, and suggested practices

as a point of reference when it comes to issues being faced in the industry, or research being needed in a certain topic.

This gives us two general sources of information on indie game development - the case study, and the gray literature analysis.

IV. Results

In this section we will detail the obtained data and the work performed on it, as well as the results extracted.

A. Interview and thematic analysis

The interview with GameDevCo was transcribed and analysed through the thematic analysis methods detailed in [11]. Initially it was loosely coded as a way to sketch out preliminary themes and patterns, which resulted in a list of codes referring to i) types of testing performed in GameDevCo, ii) contexts of discussion - present, past or future practice, and iii) highlights of problems faced.

After scanning through these codes, 29 sub-themes were noted down, then spread out into a mindmap, stemming from their logical relationships to each other, and naturally forming clusters of similar subjects. These clusters were then used to formulate the four final themes, which i) summarised the sub-themes, identified as belonging to each cluster, and ii) could together serve as a condensed summary of the entire interview. However, as expected by the thematic analysis guidelines, not all sub-themes could relate to large, final themes. Instead, they were kept in mind as context for the results.

The final themes were as follows:

T1: A game's final success depends on the ”fun-factor”. In the case of indie games, a game's marketing opportunities are closely related to online streamers' coverage [12], and thus the game's ability to entertain stream viewers[8].

T2: As game success is influenced primarily by its quality-in-use (fun-factor), the development process prioritises gameplay design quality over the technical execution - a fundamental difference in the approach to testing, in comparison to

[8]From the interview with GameDevCo: “We need to playtest to make sure that it's fun for them to both play the game, and make a video of the game in a way that makes the game seem fun for others to play as well. “

TABLE II
LIST OF PROBLEMS AND OTHER HIGHLIGHTS, RESULTING FROM THE THEMATIC ANALYSIS OF THE INTERVIEW WITH GAMEDEVCO.

ID	Theme	Content
P1	T2	Performance reporting/crash reports are cryptic and hard to trace back.
P2	T4	Non-point-and-click games deal with floating point numbers, hard to predict with accuracy for automated testing.
P3	T4	Dependencies between development platforms (Unity) get in the way of de-coupling with the purpose of testing.
P4	T4	Need to detect logical errors/soft-locks via automated testing.
P5	T2	Testing is not seamlessly integrated in the workflow - takes up too much time to set up and write tests.
P6	T3	Playtesting feedback is often contradictory - there is no universal language and every player's experience differs.
P7	T3	Player feedback is unreliable as a single source to point out gameplay design faults.
P8	T3	Players tend to be timid with playtesting feedback.

software engineering practices.
T3: Playtesting is an exploratory way to assess the fun-factor (gameplay design) of a game, but is so far largely human-dependent.
T4: QA is quantitative and objective, but cannot be relied on for fun-factor assessment. Dedicated QA staff is impossible for companies of a small size and automation is difficult to implement.

In addition to the final themes, a valuable extract of the initial interview are the highlights of problems GameDevCo face, as well as their ideas for an ideal hypothetical testing set-up. These highlights were also fitted into clusters, based on their relevant aspects of testing, and were taken into consideration when formulating the final themes. The details of subthemes pertaining each theme is presented in table III. Moreover, A list of problems identified at the case company is presented in Table II. Each problem is connected to one of the themes resulted from our thematic analysis.

In the second interview with GameDevCo it became apparent, that after the first interview they had attempted employing some test-first development as a part of their workflow, and seen some positive results. In their words, despite not having worked with TDD-like practices for very long, they felt TDD had already brought them a level of regression testing coverage they initially lacked. GameDevCo made sure to highlight they would not be aiming for 100% test coverage, however - but that TDD gave them more coverage than they initially had either way.

B. Gray literature review

In an effort to provide a clearer, more objective answer to research questions regarding the industry as a whole, the analysis was designed to include data scraped from the web. This started with an analysis of the 100 most relevant posts on reddit.com/r/gamedev when one searches for the keyword ”test”. Each post was manually read and sorted into a category, based on the context the word ”test” was mentioned in, or the contextual meaning of the word itself.

TABLE III
LIST OF SUB-THEMES, RESULTING FROM THE THEMATIC ANALYSIS OF THE INTERVIEW WITH GAMEDEVCO.

ID	Theme	Content
ST01	T1, T4	Testing prioritisation in university - little playtesting, no QA.
ST02	T1, T4	Priority of failure fixes: number of users affected (hardware specification-wise).
ST03	T3	Testing as a means to evaluate game design and the original vision for the game.
ST04	-	Testing is spread unevenly in the development process - different types at different times.
ST05	T2	Features are implemented several times (as proofs of concepts), but flexibility isn't a priority.
ST06	T4	QA testing is outsourced - QA is not needed all year round (ST4) so commissions make sense.
ST07	T4	QA testing affects all (mentioned) areas of feedback positively.
ST08	T4	QA-found fixes and game patches are released on an as-necessary basis - depending on the platform and how finished a game is.
ST09	T4	QA reports performance issues, graphical errors.
ST10	T4	QA plays a role in releasing a more stable game (ST8)
ST11	T4	Fixing bugs is a positive thing - shows activity and engagement.
ST12	T4	QA staff positions are seen as temporary - a gateway into ”real” game development.
ST13	T3	Playtesters sometimes find bugs, but it is not their main task.
ST14	T2	Lack of systematic regression testing, feature failures are fixed as they come up/are noticed.
ST15	T4	This company prioritises product quality over product size/quantity.
ST16	T2	Bugs or faults are only treated as important if they are noticeable/there are no searches for faults which have yet to make themselves known.
ST17	T3	Playthroughs are natural - no script generated.
ST18	T3	There is structured playtesting, but developers play through the game naturally too.
ST19	T1	Game value/success/marketing is reliant on the ”fun-factor”, as well as the ”streamability”.
ST20	T1	Time spent playing is a metric for fun-factor.
ST21	T1	Only ”fun-factor”-passing games make it past the prototype stage.
ST22	T1	”Quick” indie titles have small development brackets - and no time to spend on testing extensively.
ST23	-	Tasks are defined beforehand - every week new implementations are evaluated by the team.
ST24	T4	QA tests are performed after A-testing, as a final touch-up.
ST25	T4	Outsourced QA testing evaluates performance on different hardware.
ST26	-	Weekly reviews of code are viewed as a part of the testing process.
ST27	-	Iteration meetings keep developers up to date.
ST28	T4	Future plans for unit testing of back-end.
ST29	-	The role of performance testing is to prevent bad financial situations when it comes to e.g. renting servers.

The initial purpose of this scraping was to create a list of posts for further reference, but it also provided us with a visualisation of the r/gamedev community's associations with the word. You can see this in the graph in fig. 1 - the primary associations are with QA testing and Playtesting, but other contexts make appearances as well.

This is especially interesting in comparison to the graph from the same analysis performed upon a set of Gamasutra posts. Blog posts on the platform were filtered in search for postmortems of indie game development projects, containing

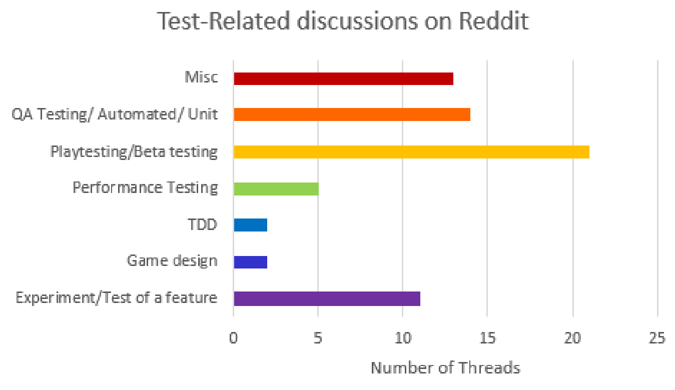

Fig. 1. Mentions of the keyword "test" in different contexts in r/gamedev. A total of 101 posts were identified, such that 33 of them did not use testing in a context relevant to our research. The remaining 68 posts discuss different aspects (i.e., categories) of testing.

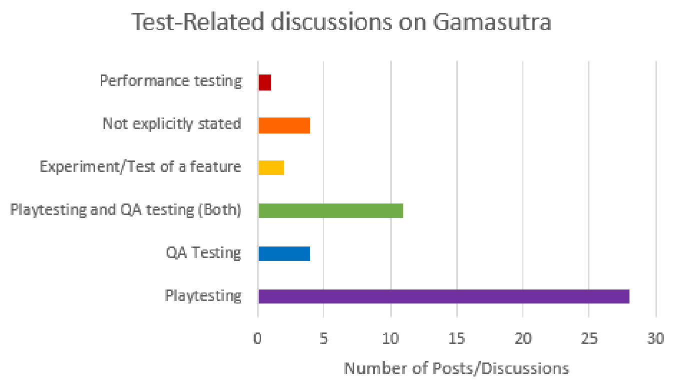

Fig. 2. Mentions of the keyword "test" in different contexts on Gamasutra. A total of 100 posts were identified, such that 47 of them did not use testing in a context relevant to our research. The remaining 53 posts discuss different aspects (i.e., categories) of testing.

the keyword "indie" and at least one mention of the keyword "test". Once scraped, the first (most recent) 100 were similarly analyzed. Unlike in the reddit collection of data, playtesting has a clear prevalence here, as one can see in fig. 2.

While providing interesting insight into the present state of online discussion surrounding indie game development, both of these sources primarily served as a pivoting point when initially formulating the interview questions, while also assisting in finding solutions to some of the problems GameDevCo was facing at the time of the first interview, and providing guidance when interpreting the game development field.

The primary source of new information, however, was GameDevCo itself.

V. Discussion

While the findings of the previous section largely confirm the information found in the preparation stages of this research effort, they have provided additional insight into the testing practices used in indie game development, particularly the practices in GameDevCo. The extracted themes highlight the difference in testing approaches, and its cause - the importance of the fun-factor - which then leads to the separation of testing into two main categories.

Understandably, considering importance of the fun-factor, playtesting takes greater priority in game development. However, the goals of that type of testing are seldom software-related. In the interview with GameDevCo, the purpose of playtesting was stated primarily as assessment of gameplay design decisions - from balance and challenge, to UX design. After explicitly asking about technical/software issues being noted via playtesting, we received an answer, which resulted in ST13 - "Playtesters sometimes find bugs, but it is not their main task".

Challenges faced in the domain of playtesting were thus centred around gameplay design assessment, and could be summarised as communication problems, stemming from the unreliability and vagueness of natural language, as well as playtesters not always being able to reliably explain what their complaints with the game are (or even not speaking up at all). In other words, the challenges within playtesting all relate to human psychology, or human interaction.

On the other hand, QA testing is primarily computer-driven and as such returns reliable feedback, which is easy to interpret. Despite this, however, it's not a priority for companies with limited resources, due to QA's disconnect from the fun-factor, the expenses of having dedicated QA staff (or having non-QA staff perform QA duties), and the difficulty in integrating automated QA testing in the game development workflow.

When questioned on the benefits of QA, GameDevCo's representative expressed it positively affects all mentioned aspects of development (Customer satisfaction, release cycle efficiency, feedback cycle speed, fewer bugs). It was also noted that minimising technical errors leads to smaller post-release patches, which keeps the developers interested (and not burnt out) in developing the game. Even with these positive effects in consideration however, QA testing is underrepresented in university syllabi on game development, and is later not prioritised as much as playtesting.

By their own definition, GameDevCo have the quality of their games as a main goal (ST15), so their solution to the common problems, which keep other companies away from systematic QA testing, was to outsource it to an external company, devoted entirely to QA testing. A scroll through the company's website shows the different services they offer - not only QA testing, but also UX and Playability testing. The company takes on games on a case-by-case basis, with individual testing packages each time, meaning there is still no permanent staff hired out.

Outsourcing QA certainly gives GameDevCo an advantage when it comes to the final technical quality, but it also creates a black box around this area of game development practices. As an example, it leads to a concentration of QA testing done all at once, once or a few times in a project's lifetime - which goes against the otherwise iterative-incremental workflow GameDevCo employs. However, since the company performs QA testing primarily as a way to smooth out the gameplay (and does not aim to achieve 100% freedom from faults), this did not seem to be a problem they needed to fix in the foreseeable future.

As it can be noted in table II in Appendix II, the self-identified problems the company faces primarily, instead had to do with their difficulties with *not* outsourcing QA, i.e. the difficulties faced when one attempts to set up their own automated testing environment.

That being said, there is no data on how many indie game development companies are in this same situation and end up leaving QA to an external company, and how many are forced to handle QA alone. Viewing other cases would make a great point for further research.

This understanding of the roles of playtesting and QA testing in game development gives us enough context to be able to review the research questions stated previously.

A. Indie games' quality assessment

Throughout the interview with GameDevCo, a primary goal was to gain as much insight into their quality assessment practices as possible. Testing in GameDevCo differs, based on the phase of development. Unlike in agile software development, QA testing is performed most commonly just once, right before the release of the game. Playtesting is performed throughout the development process, both officially - in organised sessions with external testers - and as casual playthroughs by the developers as they develop features.

However, the singular point of QA testing could be a factor due solely to the use of outsourcing - especially considering examples of other indie developers demonstrating at least partially automated testing suites[13]. Additionally, some months after the interview, GameDevCo reported they had begun implementing their own automated unit testing, covering a part of the game back-end. This serves to show the flexibility of an indie company's development process, and the relative ease with which a workflow can change - but also the vast difference from company to company.

While GameDevCo settled for making use of the integrated unit testing plug-in of the engine they work with, in addition to outsourcing its final QA tests, other smaller developers can use external tools, such as NetEase's Airtest Project[9] to perform a larger part of the automation alone. The extent and depth of testing, once again, depends on the team's budget and time pressure. Especially when taking into consideration the importance of the fun-factor for small producers.

When it comes to playtesting, most companies seem to have similar practices to that of GameDevCo - occasional scheduled playtesting sessions of an exploratory nature, with the goal of receiving feedback from the playtester on how fun the game was to play - and what could use improvements and balancing. While a decrease in effort put into QA testing would logically lead to more bugs slipping through into test versions of a game, there were no signs of playtesting being a valid, reliable and long-term substitute for QA.

In conclusion: indie games' quality assessment varies from company to company, depending on the size and experience of the teams, the development background, the financial success of the company, and the time restrictions the teams have set for their final delivery. This being said, testing follows similar patterns. Game quality is represented by i) the quality of gameplay design - the quality-in-use or the fun-factor - which is assessed through playtesting throughout the entire development process, and ii) the technical quality, assessed by QA testing .

B. Test-first in game development

While mentions of TDD (and test-first) implementations both on Reddit and on Gamasutra were notably few, there were several projects, which stood out as having not only successfully employed a TDD (or TDD-like) workflow, but also noted the benefits of it.

In a demonstration of their implementation of a TDD-like process in their recent game's development[14], a developer from Freeform Labs[10] describes one of the benefits being the easy regression testing. At the same time, no specific difference can be noticed between their development process and that of a software engineer working on a "business" application - so no claims on heightened productivity can be made from TDD alone. What TDD has helped with, has been the coverage of edge cases, which a manual tester might have difficulties replicating (or perhaps not even prioritise).

It is, however, explicitly stated, that TDD does not replace the need for qualified QA staff, partially because of the false positives it would find, due to the nature of processing 3D graphics - as mentioned in P2 of the interview (table II in Appendix II). Another reason is the more subjective quality criteria, such as the "smoothness", presence of glitches, clipping, etc.

GameDevCo did already have a functioning workflow at the time of the first interview, so none of the aforementioned downsides posed a relevant threat in their situation. Instead, their implementation of TDD-like practices provided them with the extra security in a minimal threshold of test coverage for their projects, as well as the assurance of some regression testing.

TDD approaches can thus be concluded to greatly help assure QA staff in their testing coverage, and assist in covering less likely scenarios - but they cannot completely render staff redundant.

C. Playtesting vs Exploratory testing

From the data gathered in the interview with GameDevCo, in addition to the contextual mentions of playtesting in the gray literature sources, one can define playtesting as related to - if not a type of - exploratory testing. As far as it concerns to noting down technical failures through playtesting, one could

[9]http://airtest.netease.com/

[10]https://www.freeformlabs.xyz/

even say they are the same - in templates for playtesters to fill out, descriptions of failures closely relate to those on feedback forms exploratory testers might submit.

However, with the main purpose of playtesting being the fun-factor assessment, the main problem becomes the subjectivity of the testers - the feedback becomes more qualitative than quantitative in nature. The main problem to tackle here, is human reliability.

This fundamental difference sets playtesting apart from exploratory testing, and makes it difficult to reliably compare the two. While superficially the feedback might appear similar, the challenges faced by exploratory testing [15] assume a different situation - one where knowledge of the software is an asset (unlike in playtesting), and where personal bias plays no significant role.

This puts the case of exploratory vs playtesting in a very particular position: while the methods of executing both types of testing are nearly- or completely identical, the results being sought out are significantly different, and so are the challenges met.

VI. Threats to Validity

Threats to validity are commonly grouped into seven groups[16]: conclusion, internal, construct, external, credibility, dependability and confirmability. Out of these seven, the following have posed as threats for this case study:

Internal validity - due to time constrains, we likely will not have the ability to empirically test the proposed solutions to test-related issues our subjects might be facing. As a result, there is always the risk that they will not lead to sufficient improvements. This can be counteracted through our second (and last) interview, after we have proposed them, when we will collect feedback from the company (or companies), based on their experience within game development. Empirical evaluation of our proposed guidelines can then be performed in future work.

Construct validity - the interviews involved in conducting this case study pose a big risk as the sole sources of data - a strange wording could lead to different interpretations of the questions and thus completely invalid results. To counteract that, we will be taking care in putting together the interview questions as clearly as possible. We will also always be reading them from the list (to avoid in-the-moment improvisation).

Additionally, we will be recording the interviews, in order to have the answers stated as clearly as possible.

External validity - as with all case studies, we cannot use our results to generalise for the field entirely. Any data gathered would be relevant only for the development team we have worked with. Despite this posing as a threat to validity, if caution is exercised when drawing final conclusions, and generalisation is avoided, there is no threat to the thesis validity itself.

VII. Conclusion

The games industry produces software which is written to serve, first and foremost, as entertainment. This poses its own challenges, as technical aspects of the final product become less of a priority for developers, and testing is divided into gameplay-centered testing (or playtesting) and QA testing. The difference is particularly noticeable in smaller, independent studios, whose budget does not allow for dedicated testing staff, like that of bigger companies does.

In this research paper, we interviewed an indie game development company, GameDevCo, asking about their testing practices and collecting data on any challenges they were facing. The final goal was to attempt to tackle these challenges, using tried and trusted solutions for equivalent problems, occurring in software engineering.

With the interview analyzed, we collected data from additional sources online, in order to create a better understanding of the industry environment indie games are developed in. We then answered the research questions stated earlier in this paper. According to our findings, test-first approaches did have a positive impact on the QA testing experience, but did not warrant it unneeded, and while playtesting certainly resembles exploratory testing, the two ultimately face completely different challenges.

However, these conclusions are reached under the limitations posed by the scope of this study. As research is limited in the subject of testing in game development, especially QA testing, gray literature sources were a vital source of data, which poses a threat to the final results' validity. Additionally, the interview, while sufficient as a base for exploratory research, could not be considered an entire case study on its own, and represented only one company at a single point in time.

Further research could offset this by interviewing more companies, over longer periods of time. Alternatively, one could apply a software engineering solution to a game development problem (such as a TDD-like approach to limited QA testing resources), and study its effects in an experiment, or apply ISO standards to the separate studied types of testing.

References

[1] 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, 2010.

[2] W. Barendregt, M. M. Bekker, D. G. Bouwhuis, and E. Baauw, "Identifying usability and fun problems in a computer game during first use and after some practice," *International Journal of Human Computer Studies*, vol. 64, no. 9, pp. 830–846, 2006.

[3] S. Muhammad, A. Shah, C. Gencel, U. S. Alvi, K. Petersen, U. Sattar, and A. ++, "Towards a Hybrid Testing Process Unifying Exploratory Testing and Scripted Testing," *J. Softw. Evol. and Proc*, vol. 00, pp. 1–29, 2012. [Online]. Available: http://www.bth.se/fou/

[4] T. Fullerton, C. Swain, and S. Hoffman, *Game Design Workshop: A playcentric approach to creating innovative games*, 2nd ed. Morgan Kaufmann Publishers, 2008.

[5] P. Mirza-Babaei, N. Moosajee, and B. Drenikow, "Playtesting for indie studios," in *Proceedings of the 20th International Academic Mindtrek Conference*, ser. AcademicMindtrek '16. New York, NY, USA: ACM, 2016, pp. 366–374. [Online]. Available: http://doi.acm.org/10.1145/2994310.2994364

[6] "Keynotes," in *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, April 2018, pp. 23–25.

[7] W. Trumler and F. Paulisch, "How "specification by example" and test-driven development help to avoid technial debt," in *2016 IEEE 8th International Workshop on Managing Technical Debt (MTD)*. IEEE, 2016, pp. 1–8.

[8] M. P. Eladhari and E. M. I. Ollila, "Design for Research Results: Experimental Prototyping and Play Testing," *Simulation & Gaming*, vol. 43, no. 3, pp. 391–412, 2012. [Online]. Available: http://sag.sagepub.com

[9] D. S. Janzen and H. Saiedian, "Does Test-Driven Development Really Improve Software Design Quality?" Tech. Rep. [Online]. Available: https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?referer=https://scholar.google.se/{\&}httpsredir=1{\&}article=1027{\&}context=csse{_}fac

[10] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering." [Online]. Available: https://portal.research.lu.se/portal/files/2838283/1276782.pdf

[11] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[12] M. Johnson and J. Woodcock, "The impacts of live streaming and twitch.tv on the video game industry," *Media Culture & Society*, 12 2018.

[13] J. Blowl. Highlight: Game engine programming: Fixing the fallout from a core movement code change. Youtube. [Online]. Available: https://www.youtube.com/watch?v=-wcCuOLtWmM&t=22m48s

[14] B. LoBuglio, "Welcome to rabbit hell! reliable ai locomotion with tdd," May 2019. [Online]. Available: https://www.gamasutra.com/blogs/BrendanLoBuglio/20190523/343175/Welcome_to_rabbit_hell_Reliable_AI_locomotion_with_TDD.php

[15] W. Afzal, A. N. Ghazi, J. Itkonen, R. Torkar, A. Andrews, and K. Bhatti, "An experiment on the effectiveness and efficiency of exploratory testing," *Empirical Software Engineering*, vol. 20, no. 3, pp. 844–878, 2015.

[16] A. Magazinius and R. Feldt, "Validity Threats in Empirical Software Engineering Research-An Initial Survey. Validity Threats in Empirical Software Engineering Research-An Initial Survey," Tech. Rep., 2010. [Online]. Available: https://www.researchgate.net/publication/221390199

Appendix

In otder to collect data, we design an interview instrument based on our research goal. The instrument includes an introduction section explaining the context of the research, as well as information to the participants about confidentiality and consent about their data. No personal information was collected. Below, we present a list our list of questions.

- How do you regard testing in your team, in terms of priority in your development process?
- How regularly do you test your system for faults, on average?
- Does your team "playtest"? How does your team do that?
  - Do you use any external tools for playthrough generation, do you play through the game naturally?
  - How often do you playtest?
  - What specific value (eg. Customer satisfaction, more effective release cycles, faster feedback cycles, fewer bugs after releasing patches, etc) does playtesting bring?
  - What issues have you noticed arise from playtesting?
  - Is playtesting a reliable time- and effort investment? Why?
  - How would you describe a failure during playtesting?
  - What does your team do when a failure is found during playtesting?
- Does your team conduct separate (i.e. non-playtest) QA testing?
  - What external tools do you use for QA testing?
  - How often do you run QA tests?
  - What specific value (eg. Customer satisfaction, more effective release cycles, faster feedback cycles, fewer bugs after releasing patches, etc) does QA testing bring?
  - Is QA testing a reliable time- and effort investment? Why?
  - Do you write test scripts for your game?
  - What issues have you noticed arise from QA testing? (i.e. in an ideal world, how would you have QA testing be different, if at all?)
  - What does your team do when a failure is found during a QA testing run?
- Do you perform any type of testing we have yet to mention?
  - Do you use any external tools for it?
  - How often do you perform this type of testing?
  - What specific value (. . .) does this testing type bring?
  - What issues have you noticed arise from it?
  - Is it a reliable time- and effort investment? Why?
  - What does a failure, found through this type of testing, look like?
  - What does your team do when a failure is found?
- Is there a type of testing you have heard of, that your team does *not* perform?
  - What is preventing you from performing this type of testing?
  - Are there any issues you are currently facing, which you think might be avoided (or fixed) through this type of testing?
  - Do you plan on implementing this type of testing in the future? If yes, when?