Tampere University of Applied Sciences

Developing an Easy-to-Use Usability-Playtesting Guide

Henna Ahvenniemi

BACHELOR'S THESIS
November 2024

Bachelor's Degree Programme in Media and Arts
Interactive Media

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Media and Arts
Interactive Media

HENNA AHVENNIEMI:
Developing an Easy-to-Use Usability-Playtesting Guide

Bachelor's thesis 31 pages, appendices 3 pages
November 2024

The aim of this thesis was to combine the elements of playtesting and usability testing in video games under the title of quality assurance. The existing information was old enough to require some updating, so the underlying goal was to also create a combination of methods that are the most suitable for today's video game quality assurance in small companies.

The theoretic approach focused into the themes of playtesting, different roles of a quality assurance specialist, and video game heuristics by means of concrete examples of the situations where the methods work best. Interviews were used to get first-hand information from experts on the field, and to find methods that would benefit from new and more effective approaches to the quality assurance workflow.

The results show that while an existing playtest system was viewed as convenient by all experts, there was a general lack of knowledge of alternative options. Playtesting principles and heuristic analysis were both used by playtesters, but those were seen as basic tasks of a quality assurance tester rather than separate responsibilities, which they would be in big game companies.

The two methods are essential in order to complete a product that is satisfying for the consumers, but both lack something that the other inevitably complements. Usability testing is mostly something that a user experience designer is tasked to do and that leaves a quality assurance specialist with a very different job. When time and resources are at stake, such a situation is not to be taken for granted, so this thesis combined the two careers into one effective method in order to standardise usability-playtesting for small studios. The theme could benefit from more research on the matter of method compatibility, but the direction of the combining of methods is already good.

Key words: quality assurance, playtesting, heuristics, user experience design, video games

CONTENTS

1 INTRODUCTION 5
2 QUALITY ASSURANCE 6
2.1 Assuring the Quality in Game Development 6
2.1.1 Playtesting in Game Development 6
2.1.2 Playability Testing in Game development 8
3 PLAYTESTING AND PLAYABILITY 11
3.1 Playtesting as a Method 11
3.1.1 The Process 13
3.1.2 Documentation 15
3.2 Playability Testing as a Method 17
3.2.1 The Process 17
3.2.2 Documentation 18
3.3 Playtesting and Heuristic Rules of Usability Combined 19
3.3.1 Usability-Playtesting Process 20
3.3.2 Documentation 21
3.3.3 Strengths and Weaknesses 22
4 INTERVIEWS 24
4.1 Expert interviews 24
4.1.1 Challenges 25
4.1.2 Restrictions 26
4.1.3 Solutions 26
5 DISCUSSION 29
REFERENCES 31
APPENDICES 32

ABBREVIATIONS AND TERMS

bug ticket	a bug report that includes all essential information about the bug
build	a version of a game that can be tested
collision	intersection of rigid elements within game environment
playability heuristics	usability rules or guidelines for good user interface design in games
OS	operating system
placeholder	asset that is not ready, temporary asset
QA	quality assurance
UXD, UX	user experience design

1 INTRODUCTION

Quality Assurance is a role in the game industry that ensures the quality of games so that they are ready for the markets. It is a necessary phase that often gets underrated especially in smaller or indie companies where the resources are scarce, be it budget, softwares or a team of people with enough needed skill. It is easy to think that making games is profitable and not difficult as many successful games have been made in increased amounts ever since the game industry started to grow. Making the games is definitely easy, if there are no standards. Many recent games are published lacking and broken when proper testing is neglected (Rodrigo Fernández 2023). It definitely does not make the products profitable when the experience is bad. In the worst case scenarios clients will start avoiding products from a developer if the quality can not be trusted.

This is why it is absolutely necessary to give thought to quality assurance, and the purpose of this thesis is to present an affordable and easy-to-use model to assure the quality of video games, to help small companies and independent developers make their products the best they can be with the limited resources.

This thesis presents guidelines for an effective combination of playtesting and usability testing in the same package. The objective is to create a model that consists of base heuristic rules of usability in video games and a structure for a playtesting process fitting for small-scale projects. The thesis will provide some information on useful playtesting techniques.

While the subject of playtesting and usability is wide and full of possibilities, it is not possible to cover all of them in this thesis. This is why my scope is targeted to simple solutions and an appropriate starting point for any small developer. Therefore bigger companies may not benefit from the information this thesis provides, but it can be used as a beginner material.

2 QUALITY ASSURANCE

1.1 Assuring the Quality in Game Development

In order to pass a publishers' quality requirements a game needs to be playable and mostly bug-free. A working end-product is always a result of good team work, but quality assurance in the game industry is the best way to ensure that the game is ready to be published.

QA (quality assurance) should be in charge of the playtesting workflow because they are the specialist in their area, but having a QA department in a small or indie company is not often the reality. When resources are scarce, full-time playtesters are most often the ones that are not hired. In these situations all playtesting is a shared task amongst the team members, but that is also where things get challenging. Everyone within the company has a role of their own: the designer designs the product, the programmer is in charge of the technical side, the audio artist composes music and so on. All of these specialists already have a focus of their own and it is not in the playtesting. They do not have the time to test games because they need to focus on things that cover their own scope. Only the person in the role of quality assurance has all of their time and focus on playtesting, finding ways to break the game and thinking like a player.

Simply playing a game from start to finish and confirming it works without analysing the product is not a proper way to test a game. Simply finding bugs from the system is not quality assurance. It is digging into the system, solving problems, finding the flaws before end users do and assuring the product meets all the minimum requirements of playability, contents, localisation and compatibility with the chosen platforms.

1.1.1 Playtesting in Game Development

Playtesting goes through several phases during the development process, the very first stage being stabilisation. The game goes through production testing to check out any major bugs which could either complicate or prevent QA's workflow. (Levy & Novak 2010, 57). Once the game is stable enough the real work starts and playtesters can focus on their specific tasks. There's a lot to uncover depending on the complexity of a game and every new build brings a great set of new problems to tackle. In an ideal situation the whole playtesting department is organised, the test lead distributes tasks and takes care of the output of testing team members and everyone has their own separate goals on a checklist. In smaller companies this is not often the case, for the playtesting team could consist of only one person and in some cases the creator is the sole person in the entire project. In these situations the workload suddenly increases drastically.

There are some life examples of situations where workflow has already been split between the members or sharing of tasks is heavily considered. Kalma Games CEO Atte-Petteri Ronkanen (2024) explains that while not having an in-house QA in his company, he can see that a QA in their team might end up having many other tasks outside of playtester's workflow. They would be in charge of creating and managing build uploads and moderating company shop pages. Kalma Games is a Tampere-based game company that was founded in 2024 and consists of four full-time workers. Dreamloop Games QA tester Eric Ranki (2024) became the only in-house QA in the summer of 2024. He states that previously all QA work has been the responsibility of everyone, but that resulted in inconsistent playtesting standards between the team members when everyone had a style of their own. Dreamloop Games is a small game studio in Tampere and they have made games for multiple different platforms.

1.1.2 Playability Testing in Game Development

Game companies tend to have a separate team for UX to perform any necessary usability checks, but this area of work slightly overlaps with the QA work as well. In a standard situation QA does not focus on anything else but playtesting, but there will inevitably be situations where a playtester has to analyse why their playing experience is not good. If they are not having fun, how can the end-user have fun? In small game companies UX is such an essential part of QA's workflow that it rarely is even considered a separate responsibility aside from playtesting. More often it is also a common task that everyone in the company pays attention to. Programmers, game designers and QAs can notice different problems in the usability of their game during the development, and the sooner these problems are found the sooner they can be fixed. In the long run any usability problems can become frustrating issues later on, requiring even big changes in the game that can become costly for the company.

Game researchers Jakob Nielsen and Thomas K. Landauer performed a case study to find out the most optimal size for a usability testing group and ended up to a conclusion that three people can still find usability problems effectively, but the process gets less beneficial the more or less there are people. (Figure 1, Jakob Nielsen 2000.).

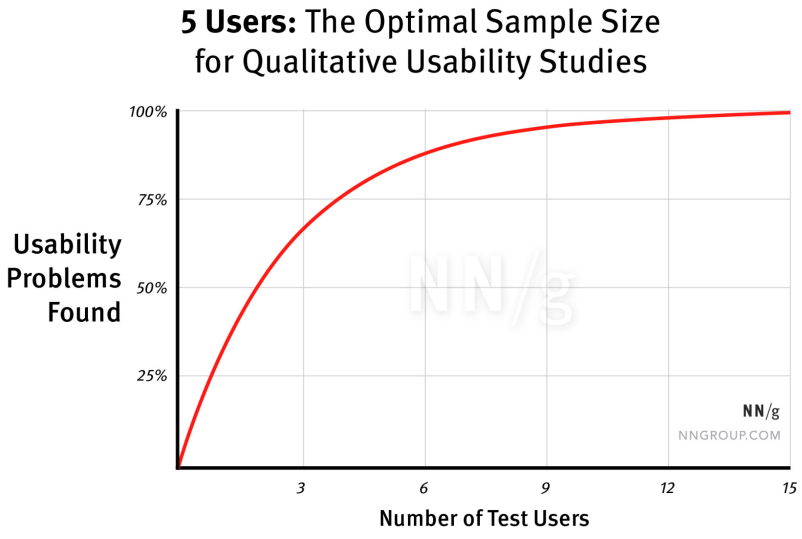


FIGURE 1. The amount of usability violations found in a product in relation to the amount of testers. (Jakob Nielsen 2000.)

To analyse the usability of a game, a method called heuristic evaluation is used. Heuristic evaluation is a key element in user experience design, and the concept has been going on for quite a long time, but ever since Jakob Nielsen and Rolf Molich (1990) published a study that contained a compact list of nine usability heuristics for user interface the full potential of the system got a firm basis (Paavilainen, Korhonen, Koskinen & Alha 2018). Heuristic evaluation of user interface is a set of rules that describe the core requirements for a good usability of a product (Table 1).

TABLE 1. 10 Usability Heuristics for User Interface Design (Jakob Nielsen 1994).

1: Visibility of System Status
2: Match Between the System and the Real World
3: User Control and Freedom
4: Consistency and Standards
5: Error Prevention
6: Recognition Rather than Recall
7: Flexibility and Efficiency of Use
8: Aesthetic and Minimalist Design
9: Help Users Recognize, Diagnose, and Recover from Errors
10: Help and Documentation

These rules should however be used in a flexible manner, as a tool of analysis instead of a truth-set-in-stone, to avoid bad design choices and to improve the quality of a product. (Nielsen & Molich 1990; Paavilainen et al. 2018).

The pros of heuristic evaluation are that unlike playtesting, heuristic analysis can be conducted early in the development of a product (Nielsen & Molich 1990; Paavilainen et al. 2018), so that playability can be improved early on. Later when a QA expert focuses on finding issues through playtesting, it is most

likely that heuristic analysis has uncovered the majority of issues with a little time and effort so the time-consuming phase of playtesting will be less costly for the game company (Paavilainen et al. 2018).

3 PLAYTESTING AND PLAYABILITY

3.1. Playtesting as a Method

Playtesting needs time and effort and hundreds of continuous playthroughs in order to find all of the most crucial game breaking bugs and most of the urgent bugs, so that the game can be fixed into a playable and enjoyable experience for the users. One QA expert can not possibly find everything alone, nor is it even efficient. According to Nielsen (1990, 2000) the most efficient number of testers is three people, as only three people are needed to find at least 75% of usability problems without major risk of unnecessary work and waste of time, which in turn increases rapidly after four or more people. The product needs to go through multiple rounds of testing after the old problems are fixed, so it's better to use resources effectively rather than trying to achieve the impossible result of a perfect problem-free product (Nielsen 2000).

When a new game build is provided by the developers, it is time to take it under the critical eye of a playtester. The first round of testing is always the most chaotic one, because the issues may be limitless. It may even feel challenging to decide where to even focus. This is why it is useful to go through all the possible issues in categories. Keeping a basic table of the basics is both a good start and a place to return to during any stage of the playtesting process (Table 2).

TABLE 2. Common bugs in categories (Levy & Novak 2010, 77).

Category	Example Bugs
Graphics	Asset is missing Wandering pixels Texture glitch
Level design and collision	Missing geometry or too big or small hitbox Invisible wall Getting stuck
Physics	Object floats unintendedly Object never stops moving Colliding sets objects flying unnaturally
Audio	Placeholder audio or incorrect audio present Volume is too loud or too quiet Missing audio
Localisation and text	Placeholder text present Missing characters Translation error
Stability	Game crashes Game freezes Loading fails
Performance and functionality	Button does not work Long loading times Low frame rate
Compatibility	Game is not compatible with intended OS Game controller does not respond
Networking	Connection cuts abruptly Lag
AI	Can not perform intended tasks Is too passive

The end goal of playtesting is to make sure that the game works as intended, meets the requirements of a chosen platform and that the audience's experience matches the developers' intentions (Miller 2014).

Every game will inevitably have unique bugs of their own depending on the complexity of mechanics, graphics and UI, so everything can not possibly be covered within a single table, nor do the bugs even appear similarly in every situation. For example, a new game update in *Deep Rock Galactic* brought a visual bug that is more likely to occur in 3D games rather than in 2D games (Picture 1). A QA expert needs to hone their problem solving skills and attention to detail to uncover these unique bugs. The model is not displaying the unit's texture right and the red shading color is visible instead when the enemy shell should be entirely black.

PICTURE 1. Graphics bug on an enemy unit in *Deep Rock Galactic*.

In the end every round of playtesting is a puzzle for the QA tester. For some this is entertaining and exciting, but others find it even funny. The harder the bugs are to find and replicate, the more problem solving skill is required.

3.1.1 The Process

Everything starts from the game document, which is usually compiled by the game designer or game director. The document lists all the essential information regarding the game, including description, mechanics and list of

assets. These are the core details a playtester needs in order to make a testing plan, and most importantly, to get an idea of the big picture. Team members should ask questions in this phase to clarify any uncertainties that they might have about the game.

The developers can be asked to provide a list of elements that require testing. This list should include known bugs and features that are not supposed to be interpreted as bugs. With clear instructions the playtester will not waste time testing features that work as intended.

Stability testing is the first huge phase in the playtesting process. In order for QA to do their job properly the game needs to be stable enough to be tested for enjoyability. For example, if the flow of the game is disrupted by a game-breaking bug it is impossible to continue testing that game mechanics are in balance before the issue is fixed.

Once the game is stable enough to be played, the testing continues in a more organised manner, one area of focus at a time. Each game genre has a specific element, mechanic or problem that may not be present in other genres at all. Multiplayer games need to be tested with connectivity issues where single player games do not. Shooting games need to be tested with weapon balance while games like *Tetris* do not even have weapons implemented. Multiple test runs may be conducted along the way depending on the amount of the content, new added features and any big changes in the game. The product is ready to be published only after the QA personnel gives the green light. The game does not need to be perfectly polished, but it has to pass standards that either the director or publisher have set for the developers. Usually these standards are a company secret. Ranki (2024) says that in order to use Xbox and PlayStation developer kits one has to make an account, and follow the terms and conditions of the publisher. They are very secretive on the matter.

Playtester’s workflow can be made more easy with the proper tools. Any kind of screen capture applications like Open Broadcaster Software (OBS) can be used to record gameplay material. Many game engines have integrated profiling tools that can be used to show frame rates, computer memory and other essential

details about a chosen platform's performance. When a screen capture device and profiling tools are combined, QA can easily deliver programmers details of lag and crashes. Bugs need to be also documented, and the best way to do this is to have a system for bug ticket repository that is accessible to all team members. Especially QA and programmers need to be able to view a buglist and see all new changes at all times.

3.1.2 Documentation

Precise documentation is essential in order to deliver the information to developers in a way that helps them solve the occurring issues. Bad documentation forces developers to do extra detective work they are not supposed to do and eventually it will lead to a situation where the tester is not taken seriously at all.

A proper way to document a bug is essentially a clear one, that gives all the necessary information for the developer. If they need to ask for more details or understand the documentation wrong, the information is not good enough. A clear bug ticket report consists of

- scene
- severity of the bug
- platform and build version
- name of the tester who found the bug
- clear title of the bug
- detailed description of the bug
- screenshot and video capture. (Interviewee 1 2024; Hämäläinen 2024).

Scene is the time and place where the bug occurs. A scene can be the title screen, log-in screen, a specific level within the game or any type of menu. By providing correct scene information in the bug report a programmer can locate the setting of the bug fast.

Severity of the bug means how crucial it is to fix the bug in order to make the game work as intended. The most common method is to divide bug severities

into minor bugs, moderate bugs and urgent bugs. Minor bugs are minimal issues within the game, that may be only cosmetic. They do not break the game or break the flow of the game, but can be unfortunate. Most minor bugs do not usually do harm to the playability even when they appear in a published version of a game. Moderate bugs describe issues that are more noticeable. Moderate bugs do not break the game, but those can affect the gameplay experience significantly like a missing asset or wrong audio file. Urgent bugs are anything that makes the game unplayable. Crashes, freezes and missing assets are only a portion of all of the possibilities. Urgent bugs should always be prioritised over everything else.

Platforms and build versions are not always required to be listed, but for QA it is essential. Both build version and platform information help the playtester to redirect their focus on device-related problems or to pinpoint the first appearance of a returning bug.

The name of the tester is good to be found from a bug ticket so that either QA or programmers will know where they can ask more details regarding the bug that was reported. No bug ticket is ever expected to be perfect, but being able to contact the right person in times of need solves many uncertainties fast.

The title of the bug should tell exactly what the bug is about. It needs to be informative and appropriate rather than a joke. Any unclear information only slows down the process of fixing the bugs, so the less anyone needs to ask defining questions the better the title is.

Detailed description explains everything that is important to know about the bug. The detailed description can include what was expected to happen in a scene, what happened instead and any other information that is relevant. Bug reproduction steps should be provided in this section so that the programmer knows what they need to do in order to replicate the bug on their device.

Screenshots and video captures are the best form of showing the occurring issues on screen. Problematic areas can be highlighted with a color to show the exact spot from the visual material.

Bug tickets can be filled with any other information that meets the company's needs. The best way to make a working ticket structure is to communicate with work mates and clarify what is the information that aids every participant the best.

3.2. Playability Testing as a Method

In the process of playability testing the focus is not anymore on the bugs, but the usability and playing experience of the game. Heuristic playability rules are often used as an efficient tool and as a checklist of a kind to make sure that the game meets all the minimum playability needs for users. All of the rules are not meant to be followed religiously since they work merely as guidelines, and sometimes the rules are being broken intentionally. In these situations it is important to define what kind of features are meant to stay and which would make the player's experience only worse.

Playability heuristics come in many forms and existing rules have evolved over time to suit for specific purposes. The ever growing game industry changes constantly with improving technology, and playability heuristics just improve alongside the changes.

3.2.1 The Process

Depending on the list of heuristic rules used a team of testers is divided for each rule section that needs to be checked. Testers analyse the game at hand with one specific rule at a time in their mind and they document their findings accordingly. In this phase a tester should produce a vast amount of issues of anything that comes to mind. After a set amount of issues have been found the tester chooses the biggest issues from among the findings and those are then presented to the team.

As an example, Hannu korhonen (2016) created a playability heuristic chart for mobile games. It includes many important aspects that make a mobile game enjoyable to play, but the same rules can be applied to many other types of games with some changes (See appendice 1).

Everything always depends on the project at hand. Every game should be evaluated differently, because in the end unique systems and mechanics have unique details.

3.2.2 Documentation

An issue card is created with a simple structure that consists of a violated heuristic code like “GP1”, location of the issue, description of why the heuristic code was violated and a suggested solution to the problem (Figure 2). The stage of the severity should be clearly visible as well and a screenshot or other image has to be attached to the card.

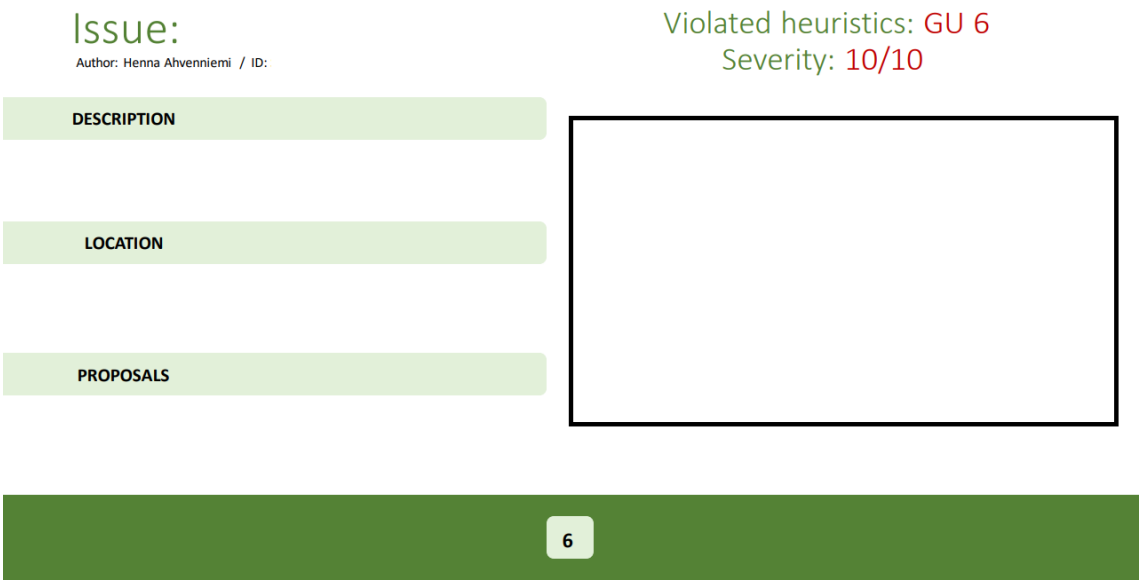


FIGURE 2. A general structure of an issue card.

It is a good habit to create multiple different cards at a time and pick out the most crucial issues. The focus of improvements will be in those very issues and the playability can be improved.

3.3. Playtesting and Heuristic Rules of Usability Combined

While playability heuristics and playtesting share the similar concept of finding issues within a product, the two terms have a very different focus in the end. Where heuristic evaluation is a tool of analysis, playtesting is a method to uncover errors in the system. The results from heuristic evaluation will not and should not always be about the negatives of the product, but also to point out what is good about it for the purpose is in addition to finding the problems to also prevent useful features being removed (Paavilainen et al. 2018). The same can't be said of playtesting that has the sole purpose of finding out anything that is wrong with the product.

Usually QA testing and UX testing are conducted by different teams, but if time and resources are scarce, it is highly practical for anyone working with these topics to have skills in both areas. Seeing the game with the eyes of a consumer and outside the box goes with both methods, but seeing both bugs and usability issues simultaneously is challenging. With a small-scale usability-playtesting model this can be achieved, however. Grouping usability issues and bugs next to each other in logic sections helps QA testers to keep in mind both sides of the testing process. One method sometimes solves the other completely, therefore making problem solving easier (Table 3).

TABLE 3. Common bugs in categories. Usability heuristics are referring to 10 core usability heuristics for UI (Nielsen 1994).

Usability Heuristic	Related Bug Examples
Visibility of System Status	Connection error Loading fails
Match Between the System and the Real World	Button does not work Placeholder audio or incorrect audio present

Usability Heuristic	Related Bug Examples
User Control and Freedom	Player can move out of boundaries Player cannot access areas Invisible walls Player gets stuck
Consistency and Standards	Placeholder asset present Translation error
Error Prevention	Game progress is lost
Recognition Rather than Recall	Menus do not open or close Assets block view
Flexibility and Efficiency of Use	Player can access things they should not be able to access Customisation does not work
Aesthetic and Minimalist Design	Asset is missing Placeholder text present
Help Users Recognize, Diagnose, and Recover from Errors	Player progress is not saved
Help and Documentation	Tutorial crashes the game

Using the heuristic evaluation system alongside playtesting can make the testing process even enjoyable. Sometimes games do not have enough content to test for bugs, but usability can always be tested.

3.3.1 Usability-Playtesting process

The process starts at the very beginning where there is only an unfinished sketch of a game. Important usability issues are being pointed out from the start so that unnecessary elements can be scrapped before they turn into waste of time for the whole production team. Because big-scale gameplay mechanics and complicated concepts require more playtesting, and bug types will significantly accumulate. Time for the entire playtesting process should be reserved for at least a month, even two.

Once the Game is fully developed the QA-testing phase begins and all urgent bugs will be fixed from the system. This will take time, but once the game is actually stable the final phase of playtesting can be started.

The third phase of playtesting is all about balance testing and checking that everything in a checklist is done and works, and publishers won't send the game back telling them that it is not ready. This is the most crucial part for compatibility QA and localisation QA because final touches will impact the audience's response. Will the game sell or will it be refunded? The game's success is not only about the absence of bugs, but playability and enjoyment as well. Video games are entertainment after all.

3.3.2 Documentation

Documentation can be done via tickets or any other bug directory list. There should always be a tag whether the issue is an enhancement or a bug, because the way these two elements are approached can differ greatly. Bugs rarely need input from anyone but programmers in order to fix them, but reports that concern enhancements should be discussed with the production team and game designer. They are in charge of the game content and therefore the correct personnels to plan any necessary changes to the game content. The changes should always be discussed together with the whole team and made sure that everyone is on the same page. Arranging a meeting is the fastest and most efficient way to get in touch with everyone.

Bug and enhancement tickets do not need to be complicated in their structure. Something that is simple enough to follow and simple enough to fill is enough (Table 4).

TABLE 4. Example of a ticket for usability-playtesting.

Section	Content
Type	Bug or enhancement
Scene	Where the bug appears
Severity	Minor, moderate or urgent
Tester	Who found the bug
Bug title	Short and clear description of the bug
Description	What happened? What was supposed to happen? What steps did you take to make the bug appear? How often did you manage to reproduce the bug? (1-5/5)
Platform and build version	Apple, Android, PC… Build version
Screenshot or video capture	Visual material of the bug

These are the most common sections that a good bug ticket should include. If anything important is left out or the information is unclear, a programmer may have to spend unnecessary time asking details that should be found from the ticket. This may happen if a bug ticket is used as a personal note, but if the bug ticket is forgotten and found later, any unclear information will tell nothing to anyone.

3.3.3 Strengths and Weaknesses

The structure is not exactly very broad and there can never be a list that covers all necessary details. The rest is left in the hands of a playtester. On the other hand it leaves enough space for improvement and all the gaps will most likely be filled by the team members according to everyone's specific needs. Audio specialists may wish to focus more on audio-related issues and 2D artists pay

more attention to the visuals. The model can evolve and create possibilities for a more sophisticated or efficient system that works for the majority of the team.

This type of system needs to be learnt to use. Every individual needs to be trained for the system and adaptation takes time. There should still be some time to be put aside for the process so it does not affect existing deadlines much, if at all.

4 INTERVIEWS

4.1. Expert interviews

The game industry isn't the same for every company. What works in big teams does not necessarily work in smaller teams. To gather more information about small game companies, four experts on the field gave their input to a series of interview questions (See appendice 2).

Interviewee 1 is a QA tester in a moderately sized company. They requested anonymity due to personal preferences. They have previous experience from a bigger game company before moving to a smaller one. They did not start originally as a QA tester, but got their job title changed from one to another.

Jani Hämäläinen is a QA tester in Kuuasema, that is located in Helsinki. He has been in the field of game development since 2005. Hämäläinen started as a 2D-artist, but continued as a QA tester at Kuuasema since 2018. The company consists of approximately 40 employees.

Atte-Petteri Ronkanen is the CEO of Tampere-based game company Kalma Games. He started his career as a junior programmer and later became lead programmer before founding Kalma Games. The company is very small and consists of four full-time employees.

Eric Ranki is a QA tester in Tampere-based Dreamloop Games. He changed career into the game industry in the summer 2024 through a work trial and became a sole QA tester in the team. Dreamloop Games is a small 25-employee game company.

4.1.1 Challenges

Time is what QA testers need the most in their work and yet there often is barely any. The interviews revealed the most time-consuming elements to be communication challenges, bug replication and testing of small different variations within the game. Interviewee 1 (2024) stated that when their own team and an external QA team communicate on different platforms it takes a lot of time from them to share information mutually. New builds are being done daily, and staying on track of the findings of two separate teams is complicated even with a common platform. Interviewee 1 also said that there is way more work than there is time, but the reason is entirely that there are not enough QA colleagues who could share the workload with them. The release schedule should be put into proportion with the size of the team and the complexity of the game project, because proper testing takes time.

Hämäläinen (2024) noted that even good plans do not always work as intended because problems are inevitable in the game industry. Something will always go wrong at the most crucial time even when the schedule is seamless. Fixing some big problems can even take a whole day because sometimes an unexplainable bug prevents the game from running even though it was supposed to be working. A change in the system can even be something small, and it still breaks something important. On that matter every new build should be tested before it is published. (Ronkanen 2024).

One QA tester can do only so much alone. If there's multiple projects going on at the same time within the team, the QA tester will be needed in all of them. The same goes with every other member in a team even if they are not necessarily playtesters. Ranki (2024) said that

> I do think that how stretched apart some of the co-workers are really dictates how communicated they are. If one of my co-worker is on 4 projects, he or she is not going to be the most responsive, because there may be people talking to them from other projects.

He also mentions that workflow gets really inconsistent easily the more there are different projects, because each project proceeds at a different pace.

Sometimes there may be no tasks to do at all and sometimes each project needs assistance at the same time.

4.1.2 Restrictions

Clients may provide only so much time for a company and those timely restrictions need to be followed. Clients may also have their own plans regarding publishing. Ranki (2024) mentions that sometimes the last round of testing is conducted by clients' own internal QA team and they make the decision to publish the game.

There may not be enough money to hire QA testers. Automated testing is not even an option for small companies even though the thought of it is tempting. However this means that someone on the team has to implement it and most often this is left to a programmer who is already busy with their personal tasks. Interviewee 1 (2024) thinks that implementing an automated testing system for a small company may not be worth the time and effort because the games are on a whole different level than big company games with much more complicated systems. People should also be trained to use the automated testing system, and there may not be enough time for that.

4.1.3 Solutions

In order to get everyone on the team on board, information should be spread so that everyone is familiar with the correct working methods, deadlines, bug findings and changes. Regular tasks should have open guides that everyone in the team can read rather than ask, but communication should also be seamless. Getting together with the team speeds up playtesting and bug fixing greatly, because bug tickets can then be analysed together and information can be shared all at once with everyone.

Proper scheduling to ensure that there’s enough time for everything. Playtesting is often such a passion career, that working overtime is a dangerously common

phenomenon according to Interviewee 1 (2024). It also lowers the risks of having to rush the playtesting at the last minute. The less there are unexpected problems or bugs at the time of the final week before publishing, the more time there is to fix everything that is necessary. An achievable scope prevents developer teams from burning out. The smaller the company is, the more harm a big project does. The more complicated and ambitious a game is, the bigger the team it requires. Sometimes small is effective.

There should always be some kind of guideline for testing. Even if there's no QA in the team, everyone can still participate in testing and everyone should participate in testing. There will be as many opinions as there are people, so it is not a possible or even recommended scenario in huge 400-employee game companies. However, it will become a strength in smaller companies. (Hämäläinen 2024). Of course everyone should also know how to do the testing. Hämäläinen (2024) mentions that nothing is more frustrating than reading badly written bug tickets. People may be hesitant to accept changes in the system, but providing co-workers easy-to-follow guidelines for bug reporting is already a good approach. Ranki (2024) recommends even taking these suggestions to higher-ups in the team first so they can spread the information to the whole team more efficiently. They have more authority to standardise new systems than a mere playtester.

A release checklist should be made an open-access material, that does a favour to everyone. Anyone can fill in things that are only important for their specific company or team in order to ensure that the game meets the company standards before it is ready to be published. The list would look something like this:

- Game is playable and there are no major bugs.
- Game has got all assets implemented.
- All of the game assets are finalised.
- Game is compatible with intended platforms.
- Localisation is correct.
- Game looks good.
- Game UI is pleasant to use.

- Content of the game does not include unwanted elements like explicit or offensive material, unless it is intended.

AI (artificial intelligence) is something that most of the interviewees were interested in. After all, using automated testing or testing with bots instead of manual testing makes repetitive work easier and saves time in big companies. It makes sense that a tool like that would make the life of playtesters easier. But the reality is that the projects in small companies tend to be so small, that using AI is most of the time not worth the time and effort (Interviewee 1). People within the company would need to be trained to use automated testing, and usually the person who implements the system into their platforms is the programmer. Since small companies should make only small-scope games, the code and mechanics in such games are rarely too complicated for a human to test manually.

5 DISCUSSION

QA testing is a wide subject, but in the end it can be turned into an efficient combination of playtesting and usability testing. There are enough similarities within the two areas of expertise to combine the essential goods into a productive working method. There is not much room to go into great details in the system, but it definitely speeds up the whole process of testing and makes error spotting much more systematic than it otherwise would. It is helpful in times when the production team is small and the schedule is tight. Playtesting is still a very time consuming process and requires a lot of thinking, but it is also a skill that can be continuously honed.

The general issues that the usability-playtesting model has are, that it is not suitable or even helpful for big companies where everyone already has a specific QA testing expertise and the UX team takes care of the usability side alongside many other things that this thesis did not even cover. Another issue is that the scope of the model does not include any specific details of rare and situational bugs or heuristics and leaves it up to the tester to figure those out. It may lead the playtester's focus to too specific things and prevent creative outside-of-the-box thinking from a playability point of view.

Further research is needed to create an even more functional system that would give space for creativity and analytical thinking. There's also space for trying out more different combinations of the playability heuristics and playtesting process, documentation and continuous improvement.

It was rather interesting to notice that the interviewees did not find that they wished for things to change for them. That indicates to me that either they are comfortable with their existing methods or they just do not know what could be improved. There is not a lot of fresh information regarding QA testing in the media, so the only possible way to spread up-to-date information is to meet these people and share the knowledge. The interviewees also expressed compassion for the companies that could not make the games the best they could be before publishing, because they felt that everything depends on far too

many things in the end. Most of the reasons for buggy games may not even be something that they can physically or legally do anything about. The world of the game industry is rather merciless in the aspect that time and money dictates many things within it, but luckily that is not all there is to it. All of the small companies that survive the harsh start show that passion is not dead.

REFERENCES

Fernández, R. 2023. Review Bombs: The Problem with Buggy Game Launches. Released on 3.5.2023. Read on 8.12.2024 https://www.gamingpitstop.com/post/buggy-game-launches-review-bombs

Hämäläinen, J. Kuuasema QA tester. 2024. Interview on 1.11.2024. Teams.

Interviewee 1. 2024. Interview on 23.10.2024. Teams.

Korhonen, H. 2016. Evaluating Playability of Mobile Games with the Expert Review Method. Dissertations in Interactive Technology, Number 24. School of Information Sciences, University of Tampere.

Levy, L. & Novak, J. 2010. Game Development Essentials: Game QA and Testing. Delmar: Cengage Learning.

Miller, L.L. 2014. UX & QA Testing: What's the Difference?. Released on 2.6.2014. Read on 26.4.2024. https://www.gamedeveloper.com/production/ux-qa-testing-what-s-the-difference-

Nielsen, J. & Molich, R. 1990. Heuristic evaluation of user interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90). Association for Computing Machinery, New York, NY, USA, 249–256. https://doi.org/10.1145/97243.97281

Nielsen, J. 1994. 10 Usability Heuristics for User Interface Design. Released on 24.4.1994. Read on 26.4.2024. https://www.nngroup.com/articles/ten-usability-heuristics/

Nielsen, J. 2000. Why You Only Need to Test with 5 Users. Nielsen Norman Group 18.3.2000. Read on 26.4.2024

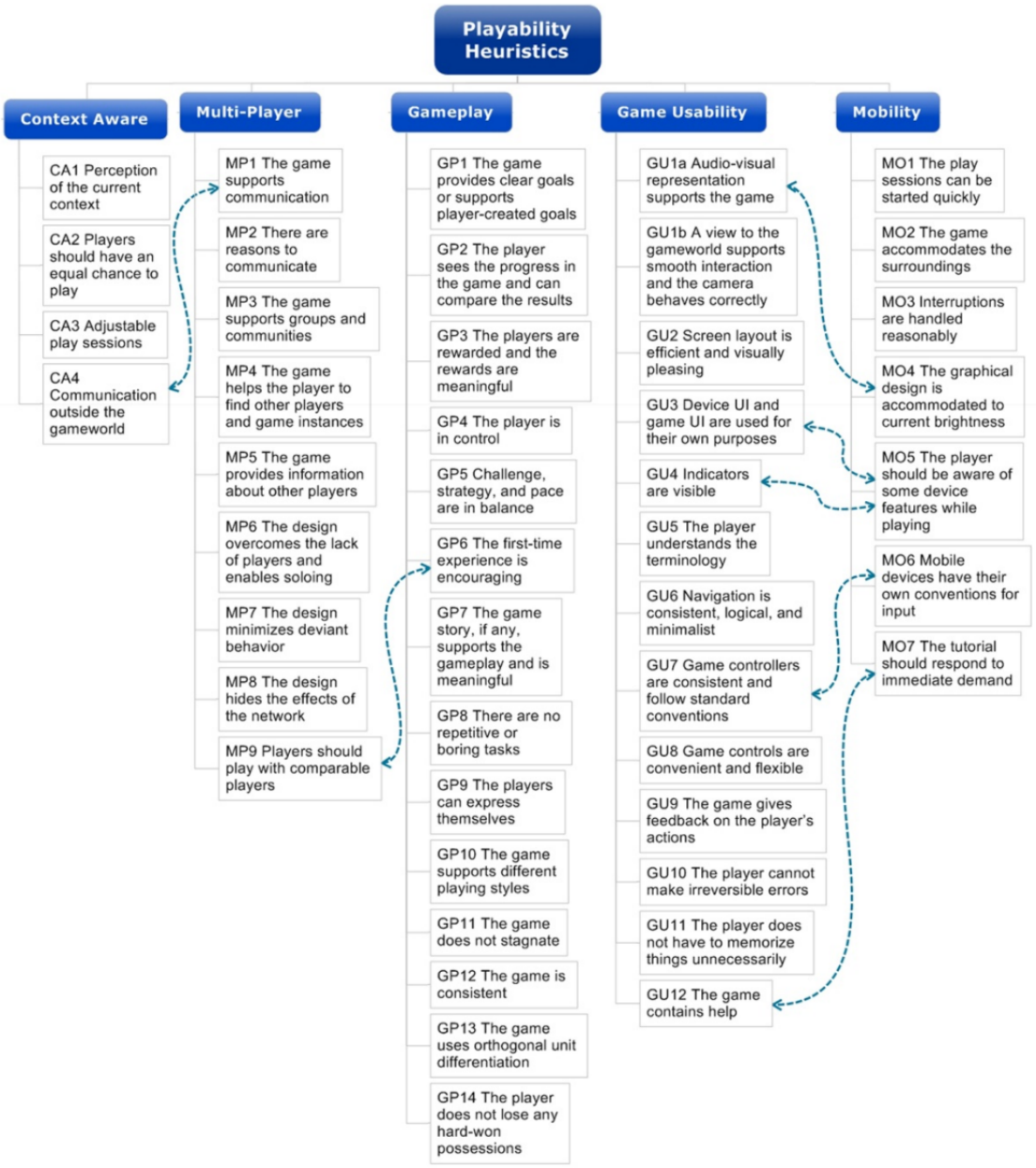
Paavilainen, J., Korhonen, H., Koskinen, E., Alha, K. 2018. Heuristic Evaluation of Playability: Examples from Social Games Research and Free-to-Play Heuristics. Oxford University Press. https://urn.fi/URN:NBN:fi:tuni-202101251674

Ranki, E. QA tester. 2024. Interview on 11.11.2024. Discord.

Ronkanen, A. -P. Kalmagames CEO. 2024. Interview on 1.11.2024. Teams.

APPENDICES

Appendix 1. Playability heuristics for mobile games (Korhonen 2016, 68).



Appendix 2. Interview questions 1(2)

INTERVIEW QUESTIONS

1. Please, tell me about yourself! Who are you, what is your job title, and what is your experience in the industry?

2. How big is your game industry team or company?

- Sole developer
- 2-5 people
- 6-15 people
- Big house with multiple, fairly organised roles (15+)
- Other (freelancer etc.)

3. How many playtest rounds are typically conducted?

4. What are the most time-consuming parts of playtesting?

5. List things you find helpful in the role of a playtester.

6. As a quality assurance personnel, what other responsibilities do you have along with playtesting?

7. What kind of struggles do you encounter in your QA duties?

8. What are your methods of documenting issues?

9. How do you know when a game is ready to be published? Are there general rules to this?

10. Do you feel that published games today would have needed more testing?

11. What is your take on automated testing?

2(2)

12. Is there something you wish would be different in the game testing process?

13. Have you ever had to change your testing routines based on company policy or some other reason apart from deadline issues?

14. How could the assurance of quality be made easier and more accessible for small game companies?