Juho-Jaakko Leppälä

The Process of Quality Assurance at Critical Force Ltd

Bachelor of Business Administration

Business Information technology

Autumn 2018

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Leppälä Juho-Jaakko

Työn nimi: Laadunvarmistusprosessi Critical Force Oy:ssä

Tutkintonimike: Tradenomi (AMK), tietojenkäsittely

Asiasanat: Laadunvarmistus, prosessi, pelisuunnittelu, pelit, pelitestaus

Opinnäytetyö on kehittämisprojekti. Opinnäytetyön tarkoitus oli kuvailla ja kehittää laadunvarmistusta Critical Force Oy:llä. Tavoitteena oli osoittaa laadunvarmistuksen ongelmat ja ymmärtää, mistä hyvät ja huonot toimintatavat laadunvarmistuksessa koostuvat. Toinen tavoite oli löytää ratkaisuja tämänhetkisiin laadunvarmistuksen ongelmakohtiin. Työssä keskitytään pääasiassa laadunvarmistuksen prosesseihin testauksen sijaan.

Tämä opinnäytetyö on tehty yhteistyössä Critical Force Oy:n kanssa, jossa tekijä työskentelee laadunvarmistuksen managerina. Critical Force on peliyhtiö, joka tuottaa mobiilipelejä.

Teoreettinen osuus kertoo pelinkehitysprosessista ja siinä työskentelevien henkilöiden eri rooleista. Prosessin osat ovat esituotanto, tuotanto, testaus ja jälkituotanto. Rooleja ovat suunnittelija, tuottaja, artisti, ohjelmoija ja testaaja. Seuraavaksi määritellään laadunvarmistusta, ja kerrotaan sen hyödyt ja haitat pelinkehityksessä. Hyötyjä ovat kyky löytää ja korjata ongelmia ajoissa, haittoja ovat ajan menetys korjaamiseen ja virheiden löytymisen aiheuttama paine työntekijöiden keskinäisissä suhteissa. Lisäksi perehdytään testaukseen pelinkehityksessä ja erilaisiin testausmetodeihin, joita ovat funktionaalinen ja epäfunktionaalinen testaus.

Toiminnallinen osuus kuvailee ensin käytössä olevat laadunvarmistusprosessit Critical Forcella käyttäen tiedonlähteinä yhtiön keskustelulokeja, yhtiön Atlassian (tärkeiden dokumenttien kokoelman) muistiinpanoja, ja vanhoja bugiraportteja. Seuraavaksi analysoidaan nämä prosessit laadunvarmistuksen teoriaan pohjautuen. Kaksi muuta yhtiön työntekijää varmistivat laadunvarmistusprosessien kuvauksen ja niistä tehdyn analyysin todenperäisyyden.

Opinnäytetyö antoi paljon informaatiota Critical Force:n laadunvarmistuksen toiminnoista, ja osoitti tämänhetkisten systeemien ongelma-alueet. Ongelmien korjaamiseksi yhtiössä kannattaisi ensimmäiseksi kehittää ajankäytön hallintaa. Tämän pitäisi auttaa joidenkin laadunvarmistuksen ongelmien selvittämisessä, ja se saattaisi vähentää kiireen tuntua, jota laadunvarmistuksessa ajoittain koetaan. Ajankäytön hallinta auttaisi myös muita pelinkehitysprosessin osia, sillä pullonkaulan avautuminen laadunvarmistuksessa kiihdyttäisi myös muiden tiimien toimintaa. Toinen asia, miten ongelmia voisi korjata, on lisähenkilöstön palkkaaminen laadunvarmistukseen. Tämä helpottaisi laadunvarmistuksen työmäärää ja lisäisi henkilöstön osaamista vielä enemmän, jolloin se ei hidastaisi pelinkehitystä yhtä paljon.

KAMK • University of Applied Sciences

Abstract

Author: Leppälä Juho-Jaakko

Title of the Publication: The process of quality assurance at Critical Force Ltd

Degree Title: Bachelor of Business Administration

Keywords: quality assurance, process, designing, games, testing

The practical portion of this Bachelor's thesis is a developing project. The objective of this thesis is to describe and develop quality assurance at Critical Force Ltd. The aim of this developing project is to highlight the problems of the project and understand what constitutes as good and bad practices in quality assurance at the company. Another goal is to find solutions to the main problem areas of quality assurance processes. The main focus is on the quality assurance processes instead of the testing.

This thesis is made in collaboration with Critical Force Ltd where the author is working as a Quality assurance manager. Critical Force is a gaming company who produces online multiplayer games focusing on mobile platforms.

The theoretical section tells about game development process and roles in game development. It also defines quality assurance and tells about the pros and cons of quality assurance in game development. Next it tells about use of testing in game development and different testing methods.

The operational section first describes the quality assurance processes in place at Critical Force using company chat logs, Atlassian notes (a collection of important documentations), personal notes and past bug reports. Then it evaluates these processes based on the theory of quality assurance. The correctness of my description of the processes and evaluation was ensured by two of my colleagues at the company.

The thesis gave plenty of information on the inner workings of quality assurance at Critical Force, and pointed out currently problematic parts of the systems in place. The first action to take to solve the issues should be looking into time management at the company in general. This should help with some of the issues quality assurance is currently facing, and it should alleviate the time pressure from busy periods. It would also benefit other parts of the development process, making it beneficial to everyone instead of just the quality assurance. Another action to take would be considering hiring more quality assurance personnel, easing the workload of current quality assurance even further.

Table of content

1 Introduction

Quality has multiple definitions. According to some, it is about satisfying specific needs that the product has. According to others, it is about satisfying the needs of the customer. Regardless of the definition, they all share some characteristics and opinions on how to improve the quality of the product. These similarities might cause one to think that there is one optimal way to improve quality, but in reality the needs of every company are completely different. Some companies might have higher need for quality than others, so their systems regarding quality must be more sophisticated as well. (Nanda 2005, 2.) For example, a company building planes must have higher standards for quality than a company making buckets.

Game development is a multilayered process, which needs teamwork from all branches of the development team to successfully finish a project. Game development process generally consists of following stages: planning, designing, producing and testing. These stages do not necessarily proceed in a linear order, but in a manner that lets all the teams of these stages interact with each other, and helps change the project based on feedback. Co-operation between these different teams is the key for creation of high quality product.

Quality management in general takes place during all these stages in different capacity. During planning and designing quality management is more a passive part of the process, where designers need to keep the quality of the product in mind while they are designing the features. If their designs do not meet their own quality standards, they will make changes until they are satisfied with it. During the actual production and testing quality management takes a more active role, and there are separate parts of the process completely dedicated to making sure the product meets the quality standard set for it.

One of the most important tasks quality assurance has is finding bugs. Bugs are errors and issues in the projects code that can cause unexpected things to happen to the software. These errors can manifest ways such as game crashing on the device, the game characters behaving in ways they are not intended to, or simply showing wrong graphics in certain areas of the game.

This thesis is made in collaboration with Critical Force Ltd where the author is working as a Quality assurance manager. Critical Force is a gaming company who produces online multiplayer games focusing on mobile platforms.

The objective of this Bachelor's thesis is to describe and develop quality assurance at Critical Force Ltd. The author has been working in quality assurance at Critical Force Ltd during last two years. The aim of this developing project is to highlight the problems of the project and understand what constitutes as good and bad practices in quality assurance at the company. Another goal is to pinpoint problem areas in current quality assurance processes, so that those problem areas can be improved upon in the future.

2 Quality assurance in game development

The theoretical background consists of what quality assurance (QA) is in general and in game development specifically. The theoretical part of this thesis deals with quality assurance testing methods and different kinds of problems quality assurance deals with. I will also go through the game development process in a nutshell.

2.1 Game development process

Game development process includes a lot more than programming and playing games. There are multiple different parts in game development that are all equally important to make sure that the game becomes as good as it can be. While each company has their own way and order of doing things, most development processes can be summed up to four phases: pre-production, production, testing and post-production. All of these phases have numerous smaller tasks that need to be done for the phase to be complete, and the project's larger success depends on the completion of these smaller tasks. (Nguyen 2014, 7.)

Pre-production is the first phase of game development process. In this phase all the features of the game are designed, and all the necessary planning for the development process is done. The most important part is to have a complete plan for all of the development process, including things such as documentation of things, estimated costs of the project, monetization of the finished product and required team size to finish the project on time. The amount of time pre-production takes can be completely different between different projects. As a rule, however, it should take something between 10 to 25 percent of the total development time. (Chandler 2014, 5.)

In the production phase the team starts creating the game. In many cases the production phase has already started before the end of the pre-production phase, since in many cases there are certain things during pre-production that require production to start. These can be things such as playable prototypes requested by the studios brass before approving the full production of the game. If everything was planned carefully during pre-production, the production phase should go smoothly. This is often not true, however. In most projects there are aspects that

were either overlooked during the planning or simply arose during the production itself. Depending on the plans made in pre-production, these can either hinder the existing plans by a huge degree, or not really at all. (Chandler 2014, 9.)

Testing phase is ongoing throughout most of the production phase, as the quality assurance team should be checking every new feature or other addition to the game as soon as it is added to the game. During this phase the quality assurance team inspects the game for any issues, such as bugs or differences from design. At certain point near the end of the development process, the main job of the development team switches to fixing all the issues that the quality assurance team finds during their testing, and making new builds for the quality assurance team to test. (Chandler 2014, 12.)

Post-production comes after all the other phases are done. This phase is mostly about wrapping up the project after it has been deemed ready to be shipped to the customers. It includes looking back at the project and learning about all the issues and triumphs during the development process, creating a closing kit for the project, and most importantly, relaxing a bit before the next project starts. (Chandler 2014, 15.)

2.2 Roles in game development

Game development usually starts with designers. Designers are the ones who plan out how the game works, what makes it fun to play, and how the game should be played. They write out the design plans that other people will need to be able to follow to make sure the game is made to be as it was intended to be. Designers themselves can be split into different roles. Some of the more important ones are map designers, gameplay designers and user interface designers. (Chandler 2014, 28-31) In the article by Korhonen and Koivisto (2006), they talk about different kinds of heuristics that should be observed in game development. Designers should look into the mobility heuristics with their producers and/or team leads when designing certain elements of the game, such as multiplayer gameplay. However, they mostly need to concentrate on the gameplay heuristics mentioned in the article, since they help designers to make sure the game is as challenging, rewarding and fun as it can be.

Gameplay heuristics remain the same no matter what platform the game is on. Prior knowledge in game design is a good thing to have when using these heuristics to evaluate gameplay. Some of the more important heuristics are: game providing clear goals, player can see their progress in the game, players are rewarded meaningfully, and the players are in control of the game. (Korhonen & Koivisto 2006, 14.)

Producers are the ones who decide how the game will be made. They take the design documents the designers have made and split them into tasks that their team can handle one by one. They are also the ones who plan out the schedule for those tasks and figure out how long it will take to have the game ready to go. Their most important task however is to ensure that the development team has all the resources it needs, and that the operation is running smoothly. There are usually producers of some kind for every team, although some teams simply have the team lead doing same sort of tasks. (Chandler 2014, 18-21.)

Artists are responsible for bringing the vision of the game alive. They are the ones who create the worlds and characters that the designers wanted for the game, and they make sure that the game looks like something that the players want to play with for long periods of time. Without the artist video games would still look like simple blocks interacting with each other. Although their job doesn't necessarily require them to pay much attention to the larger heuristics groups mentioned in the article, there are some singular heuristics in the game usability heuristics that artists should think about while doing their tasks, such as making sure audio-visual representation enhances the gameplay and indicators not blending with environment. (Chandler 2014, 22-24)

Programmers work side by side with the artists to bring the game into being. More specifically, programmers are the ones who create the actual functionality of the game. They make the gameplay work as planned, make the UI do what it is supposed to and fix the issues that the player runs into. Programming is one of the hardest tasks in game development. It can be especially difficult if the game has a lot of old legacy code that is still needed for the game to function, even if it is rarely used anymore with addition of new improved features that do the task better. (Chandler 2014, 25-28)

Quality assurance, be it testers or managers, are usually mostly used near the end of feature or project development. Their mission is to make sure everything works and looks as it is supposed

to, and report all the issues that need to be dealt with before the game can be released. They should also have the final say in if and when the game can even be released. At this point all of the heuristics should have already been looked into in some extent, but quality assurance should keep all of them in mind while testing things, and give their feedback if things feel off in some way. (Chandler 2014, 32-33.)

2.3 Definition of quality assurance

According to Test Institute (International Software Test Institute) quality assurance in software can be split in two aspects: external and internal quality. The external quality concentrates on the quality of the user experience, while the internal quality concentrates on the quality of the code the programmer creates. (International Software Test Institute 2018, 12.)

Quality assurance is a wide definition. The official definition is defined by International Software Qualification Board (ISTQB) as “Part of quality management focused on providing confidence that quality requirements will be fulfilled” (ISTQB 2018, 52). In other words, quality assurance tries to make sure that the product is of good quality in the end. Quality Assurance can be any process of control if it assures that product that is in development is satisfactory to all specified demands (Komár 2017, 5).

In game development, quality assurance has a part in all phases of development, but it is most important in the production phase (Komár 2017, 9). While this may be the case, in the whole of game development industry there is no one style of quality assurance that everyone uses. Because of this, every developer has their own way of doing quality assurance (Bethke 2003, 52). In this thesis I will describe the specific way Critical Force does quality assurance.

Figure 1. Quality control and quality assurance (International Software Test Institute 2018, 17)

Quality assurance is often confused with quality control. Many seem to think these two terms are interchangeable with each other. This is not the case however. Quality control is a smaller part of quality assurance, just like software testing is a smaller part of quality control. Quality control's main task is making sure that the project is being made in the accordance with the client's specifications. Testing the project is one of the main quality control tools. (International Software Test Institute 2018, 17.)

Table 1. describes the differences between quality assurance and quality control. The main differences between the two are in their orientation. Quality control tries to find the issues in the product after it is done, while quality assurance aims to prevent those same issues from ever happening in the first place. Quality control also affects the product specifically, while quality assurance affects the whole development process, all the different parts of the development team and all of the projects that are done within the company. (Software Testing Fundamentals 2018b.)

Testing is one of the most important parts of quality assurance in game development. Without testing it would be impossible to be sure about anything working as intended in the game. Even if a development team doesn't have quality assurance personnel in the team, they will do rudimentary testing as much as they can to be certain that their product functions in intended manner. More details about testing in chapter 3.

Criteria	Software Quality Assurance (SQA)	Software Quality Control (SQC)
Definition	A set of activities for ensuring quality in software engineering processes. The activities establish and evaluate the processes that produce products.	SQC is a set of activities for ensuring quality in software products. The activities focus on identifying defects in the actual products produced.
Focus	Process focused	Product focused
Orientation	Prevention oriented	Detection oriented
Breadth	Organization wide	Product/project specific
Scope	Relates to all products that will ever be created by a process	Relates to specific product
Activities	• Process Definition and Implementation • Audits • Training	• Reviews • Testing

Table 1. Differences between quality assurance and quality control (Software Testing Fundamentals 2018b)

2.4 Pros and cons of quality assurance in development process

Although quality assurance is a much needed and helpful resource for software development, in some ways it can also be detrimental for some parts of development. Engel (2014) describes in her article a situation where a quality assurance manager has volunteered to help other teams with certain tasks to help them save time. While this seems like a good move, in the end both the team and Engel herself end up using more time to deal with the issues that rose from Engel dealing with the tasks. Similar overly eager helpfulness isn't necessarily limited to only quality assurance. However, since quality assurance is more limited to certain parts of development, they might have more free time to tackle issues from other parts of development when they have no immediate tasks to do.

Another good example of the duality of quality assurance in game development can be found in an article written by Schreier (2017) for Kotaku. In his article he writes about how in some studios quality assurance aren't treated as equal members of the development team, since the quality assurances job is to break the game, whereas the developers job is to fix the game. Thus, they can be seen as the enemies, since they show the programmers all their mistakes. This can cause friction and stress during possibly already stressful project. On the other hand, it is good

that the issues are found and dealt with, but on the other it can grate in the mental health of the development team.

Quality assurance can also cause a lot of delays to the production if it is done to the finest detail. In his article Schreier (2017) also mentions cases where quality assurance has found tons of issues that need to be fixed, but because of the deadlines for the project, there is no time to fix all of them. Issues that do not flat out prevent the game from working may be ignored when found, since they do not prevent the release. Some other issues might be scheduled for fixing post-release, if enough people find them and start raising noise about it. Some might never get fixed, if they are extremely rare or hard to reproduce. This is often the reason why quality assurance gets blamed for buggy releases of modern games. The issues were found in time, but the schedule and deadlines didn’t allow for them to get fixed.

3 Testing in quality assurance

In this chapter I will go through what testing is, how it is used in game development, what kinds of testing there is, and what kind of testing methods there are.

3.1 Testing in game development

Testing is the main tool of quality assurance, and while all developers want to do as much testing as possible, not every studio has their own quality assurance team. External quality assurance companies are a common tool to use in game development when your own team does not have the manpower or budget for internal quality assurance team. Another way to handle testing without full quality assurance team is to automate as much of the testing as possible. (Komár 2017, 11-12.)

Game testing can be a quite stressful and daunting job. Most game testing is about repetition over and over again, over a long period of time, and that can get frustrating quite fast. Often the testers concentrate on singular part of the game so much that they have no chance to really play the game. Testing can also cause delays in schedule that projects often do not have time for, causing even more stress to those that have to report issues causing these delays. Unfortunately, this is one of the reasons many companies cut testing time if other issues happen during development. This is why testing should be scheduled into the development as early as possible, to make sure that issues do not get ignored due to lack of time. Producers and quality assurance should also discuss early on how much testing each feature is going to theoretically need, and limit certain functionalities if the testing would be next to impossible with given time. (Chandler 2014, 235-236)

In the past testing used to only be done at the end of development to save time and money for other parts of the project deemed more important than testing. These days testing is integrated throughout the development process, ensuring that the project has gone through thorough testing before it even gets to the hands of the testers. This prevents large amount of issues from ever getting through to the testing phase and eases the amount of bugfixing at the end of production. (Redavid & Farid 2011.)

One of the best testing tools is simply having regular builds of the game whenever there are changes in the game. This helps in catching issues early, since the new additions can be tested before they make it to the live version of the game. Builds are also helpful in finding any kind of visual issues on the target platform that couldn't be found on the editor on PC, since the way everything looks visually is quite different between every platform. Some bugs might also prevent the builds from compiling properly, which will also notify the team that there are issues to fix in the build. (Chandler 2014, 225)

There are multiple different kinds of testing in game development that do not all have something to do with quality assurance. In game design, designers can do tests to find out if their game prototypes are fun to play (Lucero, Holopainen, Ollila, Suomela & Karapanos 2013). This is an important step in designing the game. By the time something comes to quality assurance for testing, testing things for fun is no longer a concern. In quality assurance, tests are about making sure things function as intended, and checking for visual issues.

Testing is also done to make sure the customers find the game enjoyable by having either private or public test sessions before the game releases. These aren't a norm, but holding these sessions can be a good way to find out what parts of the game might need some more polish or adjustments before the game releases, or make even some radical changes if some parts of the game were thoroughly unenjoyable.

Another important thing to test regarding mobile games is connectivity, especially when it comes to multiplayer games. Mobile games often use the players internet connection to contact other players, the game's servers, or the app marketplace of the used device. There aren't perfect ways to test connectivity, but automated testing can be used to emulate the online functions of the game. (Helppi 2014c.)

Most testing has historically been done manually, but recently test automation has become more and more common when testing software. Automatic testing that is integrated into the development process itself can help development teams to find all the issues in the game and get them fixed before the game is shipped out. Compared to automated testing, manual testing has way more faults in it. Since manual testing is done by humans, it is more prone to miss things, way slower than automatic testing, and unable to do as detailed a test as automatic testing can. (Helppi 2014b.)

3.2 Testing methods

There are four main methods that software testing can be split into: code-based testing, object-oriented testing, component-based testing and specification based testing (Muccini 2009). All of these can be used in quality assurance testing to some degree, but specification-based testing is the main style of testing quality assurance does. In specification-based testing the tester follows given test case and makes sure the software under test behaves as the test case says when given specific inputs. If it does not behave as intended, the tester reports the behavior as a bug. (Redavid & Farid 2011.)

Black-box and white-box testing are also testing terms that can be applied to software testing. In game testing, they do not differ from normal software testing, but their end-goals are not the same. In Black-box testing the testing concentrates on the actual gameplay elements of the game, such as UI functionality, visuals or animations. White-box testing on the other hand concentrates on the functionality of systems and components outside of the gameplay elements, such as third-party plugins, engine functions and audio functions. (Helppi 2014a.)

There are many ways to do testing in quality assurance, but some of the more important testing methods can be listed into non-functional and functional testing. Non-fuctional testing focuses on testing things that do not affect the game itself directly, such as hardware and ease of use. Functional testing on the other hand focuses on testing that the game itself works as it is supposed to, with tests inspecting the smallest parts of the game and how they work together. Non-functional testing consists of vulnerability, compatibility, usability and performance testing, and functional testing consists of unit, integration, system and acceptance testing (Cordasco 2016.)

One of the easiest to understand testing methods is Ad Hoc testing. In it, there are no set steps or planned tasks to test. Instead, everything is improvised during the testing sessions. Because of this, it is also called monkey testing or random testing. It can be helpful in finding edge cases that wouldn't have been found with strict planned test cases. On the downside, found issues can be harder to reproduce, since there are no documented steps to take to end up at the issue. Often the tester will have to try to recall the steps they took to recreate the issues, which can be hard since they were doing whatever came to mind at the time of testing. Depending on the creativity of the testers, the success of ad hoc testing can vary greatly. Regardless, it is the

simplest and often best ways to find the gamebreaking issues, and it doesn’t require any kind of training or education for the testers (Software Testing Fundamentals 2018a.)

4 Quality assurance of Critical Force

In this chapter I will describe quality assurance at Critical Force Ltd. in as much detail as I am allowed to. I will first describe developments in the quality assurance during past two years based on the background given above, after which I will describe the current quality assurance practices in use at the company, and the current problems.

4.1 Starting point and the purpose of the thesis

The practical portion of this thesis is a developing project. I will analyze the process of quality assurance at Critical Force Ltd. which is the client of this thesis. The goal is to locate any problematic parts in quality assurance at the company and improve upon them.

The thesis will have succeeded if it describes accurately the development and the current problematic parts of the quality assurance at Critical Force Ltd. and manages to give possible solutions to those problems. Additionally, other marks of success are if the company considers developing the quality assurance department further.

Critical Force is a Kajaani-based mobile game company aiming to create a thriving mobile esports scene. The company was founded in 2012 by Veli-Pekka Piirainen and a group of game development students, with a vision of creating an online first-person shooter game for mobile environment (Critical Force 2018). One of the biggest reasons to concentrate in mobile games was the ever-growing market. Consoles and gaming PCs are a commodity that many people have, but they are still in a point where everyone doesn't have one. Mobile devices such as phones and tablets however can be found in any household, even in the poorest families. Critical Force has released multiple small mobile games such as Critical Strike Portable, Company of Tanks, and their currently biggest game, Critical Ops.

When I started at Critical Force, there were no prior quality assurance personnel at the company. Features were tested by the person who made it first, then by the team if there was enough time. Any bugs that were found were reported the reporting tool Hansoft, and those bugs were fixed as soon as there was time. After that, the new features would be updated to

the game, and any leftover bugs would be reported by customers in various way, ranging from customer support email to posts on Facebook and angry messages sent to the CEO. Even with all these ways bugs were reported, the player base was relatively small at the time. Due to this, many issues weren't reported in a timely manner, causing some smaller problems to run rampant for longer periods of time.

I used my personal logs as well as past bug reports, company Atlassian (a collection of important documentations), customer feedback and company conversation logs to evaluate quality assurance at the company. My analysis is grounded on theory of quality assurance, and it is based on my experiences and knowledge I have gotten during working at Critical Force Ltd as a quality assurance manager and studying at Kajaani University of applied sciences.

To guarantee the correctness of the information on the quality assurance processes, I had another member of the quality assurance team and one of the producers at the company read the thesis and asked them if the information given is right.

4.2 The process of game development at Critical Force

The process of game development at Critical Force Ltd can be summed down to the chart Figure 2. below.

Figure 2. The process of game development at Critical Force

First step in the development cycle is planning. In this step the team looks at what features are needed or what has been asked for and decide on the next features to be made. After a rough plan on what features will be added, those features are given to designers to figure out how

they would work out in the game. As they figure the features out, the designers then create the design documentation for the feature. These documents should describe the functions of the features to the finest detail, to try and make the programmer's job easier in later stages. The documents might have to be edited at later date, if some things have to be changed due to issues during development. If that happens, the designers should be the first ones to try and circumvent the issues with new design.

Next comes the pre-production. In this step producers and team leads begin splitting the feature into tasks that need to be done to finish the feature. They then schedule those tasks for programmers and artists for the upcoming sprints and start thinking about release dates based on past experience and predicted time it takes to finish the feature. When the tasks are assigned, the production step starts. In this step the programmers and artists do the tasks given to them and then combine those tasks into the feature. After the initial testing done by the programmer of the task making sure that the basic functionality of the feature is okay, the feature is given to the quality assurance team.

While quality assurance mostly takes place at the end of production at the company, it does happen in lesser extent during all of the process. Main difference is that during other phases, the ones doing the task at hand are the ones making sure the quality of the feature stays acceptable. In the end of production, the quality assurance team does the full pass of the feature for bugs and other issues. Then those bugs will be assigned to be fixed by the team creating the feature, and once they are done the quality assurance team will check the feature again. This repeats until the quality assurance team is satisfied that the feature is good for release.

4.3 Development of quality assurance at Critical Force

The quality assurance team of Critical Force was founded in the summer of 2016 with my hiring as an intern. Before this, there had not been anyone specifically for quality assurance at the company, and tasks that would normally fall to quality assurance were handled by whoever happened to be free for those tasks at the time. Even then if they got new tasks while doing the testing, they might just drop the testing to concentrate on the new tasks given. This caused

many issues to go unnoticed until they were reported by the player base to the support email. Since the player base was also a lot smaller than it is now, some smaller issues would still often be missed up to the point that we are still dealing with some tiny old graphical issues.

At the start of my internship, the tasks given to quality assurance were mostly about testing new features under development for any issues, reporting those to our bug reporting tool Hansoft and testing the live version for issues that hadn't been caught previously. Additionally, if there was time from other tasks, answering support mail and reporting the bugs from there. Since the development team was quite small at the time, there was a lot more time to handle support mail while already reported issues were being fixed. After I had gotten used to those tasks, I was also given reign over handling beta testing of one of the maps under development at the time. The testing was done with the help of volunteers from our player base. The testing lasted for little over a month, and the map released couple weeks after then end of testing after issues reported in the testing were resolved.

Near the end of 2016 the development team started to slowly grow bigger, causing the number of tasks for the quality assurance team to grow. The player base was also growing rapidly at this point. Because of this, there was less and less time for quality assurance to handle support email too, and in January of 2017 there was a support person hired to help handle the support emails. At the start quality assurance was still involved in the process, since there were a lot of bug reports coming through the mail, but with time support was completely separated from quality assurance into its own team. They technically still worked in tandem with the quality assurance, but they had much more independence on the decisions they made regarding support than previously.

In January of 2017 the company paid for a lecturer to come teach about quality assurance testing to a small part of the development team. The lectures mostly covered things that were already being used at the company to some degree, such as ad hoc testing, and some finer details about how to use those things in the company. One of the new things that were not in use yet at the time was the use of test case sheets, which is now an important part of the release candidate checkup.

Near the end of March 2017, the second member of the quality assurance team, Igor Bernardo, was hired. The dev team and the product were starting to get too big for one person to handle

alone, which is why acquiring additional personnel was made a priority at this point. After a short acclamation period where Igor was given smaller tasks to handle, we started doing tasks together, allowing us to test new builds and features in half the time it took previously. Over the past year and a half our teamwork has gotten to the point where we can handle most tasks in a swift manner.

In the autumn of 2017, Critical Ops team started using automated build pipeline for creating new game builds for the quality assurance to test. The programmers would trigger a new build of whatever needed testing, and soon enough the test version would be ready to be downloaded using Hockeyapp, an application where all the builds from the pipeline are available from. With the new pipeline whoever was creating a new feature could have something testable building in the background for quality assurance while they themselves could keep working on other tasks. This sped up development time significantly since the programmers didn't have to manually create everything for the quality assurance whenever there was something to test.

Ever since summer of 2017, there has been talks about how programmers need to investigate doing more unit testing, as well as automatizing unit testing as much as possible. During latter half of 2017 they started doing unit testing more, but we have still to achieve unit test automatization. Unfortunately, there hasn't been enough time or manpower in the teams to assign anyone into looking more closely into automatization, and it seems unlikely that there will be time anytime soon.

During spring of 2018 we finally managed to set up a staging environment. Previously when our updates had changes that might affect server or game stability in some way, we couldn't be sure about how live environment would react to those changes unless we pushed the changes to alpha or beta, depending on how many people we needed to properly test the change. If those changes then happened to break something important, and affect live players, we would have to try to fix everything while players were unable to play. Naturally this would affect player numbers and their review ratings of the game. With staging environment, we can try those same changes in a nearly identical environment, without worrying about breaking the live version of the game.

4.4 Good Practices at Critical Force

The automated build pipeline is one of the most helpful tools and practices in use at Critical Force. The pipeline automatically creates new test builds whenever the programmer tells it to, instead of the programmer setting everything up manually. This helps reduce both the amount of time the programmers have to spend not working on their tasks, as well as the time quality assurance has to wait for new builds. The pipeline is also an excellent tool for the level designers to quickly get playtest data on their new maps, since they can put one map into the pipeline, and work on another while it builds. When it is ready, they can put the other one into the pipeline and hold a testing session for the first map.

Another good practice is the use of test case sheets. Test case sheets are basically a list of tasks or things to test in a build. Most commonly at Critical Force test case sheets are used whenever there is an update coming to the live version, and when a large feature is added to the master build of the game. The former is done to make sure the update doesn't unexpectedly break anything in the live version of the game. There have been times where a new feature has worked in the development environment perfectly, but interaction with the live servers and backend has somehow caused it to break the game. The previously mentioned master build testing is done because even when the new features are made to work with the game as a whole, there is always a chance that some piece of legacy code has an unexpected interaction with the added features, causing the master build to break horribly.

The amount of play testing and ad hoc testing done on the game is also an extremely good practice at the company. Whenever there is a new feature that somehow affects the gameplay, there are play test sessions scheduled with larger test groups to make sure there have been no negative changes caused by the new feature. Usually these test groups are comprised of game designers, quality assurance team, community team members and whoever else happens to be free from artists and programmers at that moment. The programmers in charge of the feature that triggered the testing are often also present for these tests to see the possible issues firsthand. Ad hoc testing happens more often than not without any planning. Usually it simply happens because there is nothing else to do for a moment, and quality assurance can just start up the live version and play for a while. More often than not these do not have result in any

new findings, but every now and then some quality of life changes are introduced due to quality assurance asking about something they thought about during these ad hoc sessions.

4.5 Problems with quality assurance at Critical Force

Main problem overall at Critical Force is lack of manpower. There are often times when there are multiple different feature development builds to test at the same time, all of them equally urgent. At those times it can get quite hectic with only two people in the quality assurance team. Often this can result in some testing being pushed back to the following day, and combined with possible game breaking issues being found, it can result in the feature being behind in schedule.

Another problem with quality assurance at Critical Force is the quality assurance teams lack of knowledge regarding quality assurance technology. Neither of the members of the quality assurance team are trained programmers, and we cannot help with quality assurance tasks that require programming as a skill. As mentioned in chapter 4.3, programmers have had no time during the development to figure out automated unit testing. If the quality assurance team had the ability to do it, they could try to figure it out during the lulls they get in their schedule when there is nothing to test.

Time management for quality assurance is another big problem, which partly causes the first problem mentioned. As said previously, there are times when the quality assurance team is extremely busy with multiple different builds coming in for testing at the same time. This can cause a bottleneck in development Figure 3., since some parts of the development team cannot keep working on their feature until quality assurance has finished checking it out. This then might force quality assurance to try and rush the checkup and miss something.

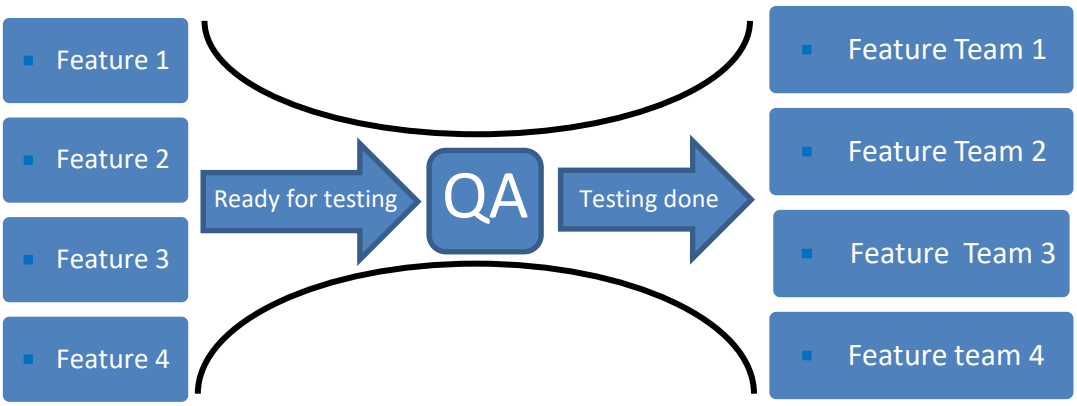

Figure 3. Bottleneck at quality assurance due to multiple simultaneous features in need of testing

To counterbalance this, there are also times where the quality assurance team goes without any meaningful work for long periods of time since there is absolutely nothing to test. These times can be seen as welcome breaks, but in truth they simply add stress to the team. It dulls concentration, and when more work arrives it can take a while to get back into full swing with the testing after not doing anything for a long while. It can also cause friction between the quality assurance team and other teams when the main development team is busy at work and quality assurance is doing nothing at all.

4.6 Solutions to problems and further development

Most of the problems quality assurance currently faces can be solved with one strategic recruitment of a knowledgeable quality assurance programmer. Both the manpower problem and the lack of technological knowledge boil down to the fact that the quality assurance team has never been properly trained for quality assurance-oriented programming or quality assurance tasks in general. With a more knowledgeable worker in the team, there would be someone consult about quality assurance methods and optimal ways to tackle certain tasks. Having a more experienced quality assurance member would also help rest of the teams to learn new things from them, making the overall experience level of the team even higher.

Another solution would be to arrange further education about quality assurance to the team in the vein of the lectures held in January of 2017, as mentioned in chapter 4.3. This would help the quality assurance team learn new and possibly better methods to use during development, and possibly even streamline the work processes currently used by the team. This could help other branches of the company too, since optimizing quality assurance would reduce the waiting time for other teams while the quality assurance are testing their current development tasks.

Another simple solution for some of the problems quality assurance team is facing would be more efficient time management for the teams. If the periods of time where the quality assurance team has too much to do could be spread out to cover the times when they have nothing to do, it would in the end speed up the development process for all the teams significantly. Currently development process can bottleneck at quality assurance during the busiest times when there are multiple things to test at the same time.

5 Conclusion

The goal of this thesis was to shed light into the quality assurance situation at Critical Force. Critical Force is a small indie game studio located in Kajaani, Finland. They wanted to improve the quality assurance at the company, which is why I created this developing project as a thesis. I mainly focused on the quality assurance processes instead of the testing, because we felt at the company that we had a good handle of testing, whereas the processes are lacking in some areas. First I described the quality assurance processes in place at Critical Force using company chat logs, Atlassian notes, personal notes and past bug reports. Next I evaluated these processes based on the theory of quality assurance. The correctness of my analysis was ensured by two of my colleagues at the company.

The thesis gave plenty of information on the inner workings of quality assurance at Critical Force, and pointed out currently problematic parts of the systems in place. The main results I got from the thesis can be put into a SWOT analysis in the following manner Figure 4:

Figure 4. SWOT-analysis of the process of quality assurance at Critical Force Ltd

As Figure 4 shows, the main strengths are all things that factor into the game testing. Automated build pipeline gives the testers easy access to the newest test versions, and test case sheets help in knowing what to test in those builds. Both of these also help in the last strength, the large amount of testing done at the company. The opportunities on the other hand are all with the personnel. Further education about the quality assurance systems and time management could strengthen the current personnel significantly. Hiring more quality assurance personnel on the other hand would ease the workload on the current staff, lessening the need to further education.

On the negative side, the weaknesses are also mostly faults in the personnel. The lack of programming knowledge can hinder the quality assurance team with some tasks, and asking for help with those tasks from programmers will cause both them and the programmers to lose time from other important tasks. This time loss will then cause more tasks to arrive for the quality assurance team before the previous tasks are done, causing the overlap in tasks. The threats shown in figure 4 are in fact mostly caused by this overlap. Development is delayed while quality assurance goes through the tasks one by one, and the overlapping tasks can cause burnouts if the amount of tasks get too high.

The first action to take to solve the issues should be looking into time management at the company in general. This should help with some of the issues quality assurance is currently facing, and it should alleviate the time pressure from busy periods. It would also benefit other parts of the development process, making it beneficial to everyone instead of just the quality assurance.

Even better documentation of my early times at the company would have been helpful for the creation of this thesis. Better balancing of work and studies would also have helped in the completion of the thesis. If I wanted to continue research even further, I could give my colleagues a questionnaire or interview them about what they see as the problem areas of quality assurance.

I deepened my knowledge of quality assurance theory, helping both myself to do my work better, and the company to get better results with their quality assurance. This new theoretical knowledge helped me to see my tasks in a new light, and helps me to develop the quality assurance at the company even further.

The company now has additional information about the issues quality assurance faces, as well as suggestions on how to handle those issues, allowing them to take action to solve them. As the person responsible for the quality at the company, I wish to also resolve problems in other areas of the development process that other development team members face. Hopefully this research will also help uncover some of those issues.

List of references

Bethke, E. (2003). Game Development and Production. Texas: Wordware Publishing, Inc.

Chandler, H. M. (2014). The Game Production Handbook. 3rd Edition. Massachusetts: Jones & Bartlett Learning.

Cordasco, M. (2016). What QA Testing Methodologies Are There? MyCrowd, A QASource Company. 12.5.2016. Retrieved 28.7.2018 from https://mycrowd.com/blog/what-qa-testing-methodologies-are-there/.

Critical Force (2018). Company's website. Retrieved 30.9.2018 from https://criticalforce.fi/presskit/.

Engel, K. (2014). Can Test Managers Harm Product Quality by being Too Helpful? Testing Trapeze 14.2.2014, 10-13. Retrieved 18.8.2018 from http://www.testingtrapezemagazine.com/wp-content/uploads/2015/10/TestingTrapeze-2014-February.pdf.

Helppi, V-V. (2014a). Mobile Game Testing – Part #1: The Importance and Difference from App Testing. 27.8.2014. Blog series of 10 Best Practices in Mobile App Testing. Bitbar testing. Retrieved 30.9.2018 from https://bitbar.com/mobile-game-testing-the-importance-and-difference-from-app-testing/ .

Helppi, V-V. (2014b). Mobile Game Testing – Part #2: UI and Functionality + Image Recognition. 3.9.2014. Blog series of 10 Best Practices in Mobile App Testing. Bitbar testing. Retrieved 30.9.2018 from https://bitbar.com/mobile-game-testing-part-2-ui-and-functionality-image-recognition/ .

Helppi, V-V. (2014c). Mobile Game Testing – Part #4: Testing for Connectivity. 17.9.2014. Blog series of 10 Best Practices in Mobile App Testing. Bitbar testing. Retrieved 30.9.2018 https://bitbar.com/mobile-game-testing-part-4-test-for-connectivity/.

International Software Test Institute. (2018). Software testing revealed. Training book. Web document. Retrieved 27.7.2018 from https://www.test-institute.org/What_is_Software_Quality_Assurance.php.

ISTQB. 2018. Standard Glossary of Terms used in Software Testing version 3.1. Web document. Retrieved 27.7.2018 from https://www.astqb.org/certified-tester-resources/glossary-software-testing-terms/.

Komár, M. (2017). Quality Assurance in Game Development. Master's thesis. Masaryk University, Faculty of Informatics. Retrieved 28.7.2018 from https://is.muni.cz/th/tlkuv/Quality_Assurance_in_Game_Development_-_Matej_Komar.

Korhonen, K. & Koivisto E. M. I. (2006). Playability Heuristics for Mobile Games. Conference Paper, January 2006, uploaded on 10 March 2014. Retrieved 18.8.2018 from https://www.researchgate.net/publication/221270478_Playability_heuristics_for_mobile_games.

Lucero, A., Holopainen, J., Ollila, E., Suomela, R. & Karapanos, E. (2013). The Playful Experiences (PLEX) Framework as a Guide for Expert Evaluation. DPPI '13 Proceedings of the 6th International Conference on Designing Pleasurable Products and Interfaces, Newcastle upon Tyne, United Kingdom 3.-5.9.2013, 221-230. doi: 10.1145/2513506.2513530.

Muccini, H. (2009). Software testing: Testing new software paradigms and new artefacts. The Wiley Encyclopedia of Computer Science and Engineering, 2716-2732. Retrieved 23.9.2018 from https://pdfs.semanticscholar.org/6049/54b82254deef726a93f7565285bba975aea0.pdf.

Nanda, V. (2005). Quality Management System Handbook for Product Development Companies. New York: CRC Press. Retrieved 27.10.2018 from https://books.google.fi/books?id=guizsuAAyR4C&pg=PA1&redir_esc=y#v=onepage&q&f=false.

Nguyen, P. (2014). Game Production and Role of Game Producer. Case study: Research on demographic of gamers in Hanoi, Vietnam. Savonia university of applied sciences, Degree Programme in International Business, Social Sciences, Business and Administration. Bachelor's Thesis. Retrieved from http://urn.fi/URN:NBN:fi:amk-2014120318055.

Redavid, C. & Farid, A. (2011). An Overview of Game Testing Techniques. Retrieved 29.7.2018 from https://pdfs.semanticscholar.org/4361/a1882ca8ea296ff6411dbbaa90ca5fbc3ed4.pdf.

Schreier, J. (2017). Quality Assured: What it's Really Like To Test Games For A Living. Kotaku – The gamers guide, Gizmodo Media Group. 18.1.2017. Retrieved 16.9.2018 from https://kotaku.com/quality-assured-what-it-s-really-like-to-play-games-fo-1720053842.

Software Testing Fundamentals (2018a). Ad Hoc Testing. Web Document. Retrieved 27.10.2018 from https://softwaretestingfundamentals.com/ad-hoc-testing/.

Software Testing Fundamentals 2018b. Differences between Software Quality Assurance and Software Quality Control. Web Document. Retrieved 27.10.2018 from https://softwaretestingfundamentals.com/sqa-vs-sqc/.