MESG

MESTRADO EM ENGENHARIA DE SERVIÇOS E GESTÃO

Analysis and Improvement of Quality Processes Inside the Betting Industry

Daniela Duarte Falcão Dias Ferraz Alves

Master Thesis

Supervisor at FEUP: Prof. António Carvalho Brito

Supervisor at Blip: Engenheiro João Ribeiro

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

2021-09-13

Resumo

O presente documento foi elaborado no contexto da tese para a obtenção do Grau de Mestre do Mestrado em Engenharia de Serviços e Gestão da Faculdade de Engenharia da Universidade do Porto. O objetivo deste projeto foca-se na análise de um caso de estudo relacionado com os processos de garantia de qualidade na indústria das apostas. Assim, de modo a que a empresa em estudo atinja uma posição firme num negócio exigente do ponto de vista da engenharia de software, nomeadamente o software de apostas *online*, a empresa desenvolveu uma nova aplicação. Deste modo, esta tese apresenta um caso de estudo focado dois processos da empresa, sendo estes os processos de desenvolvimento de *features* e também o processo relacionado com a libertação de software para os clientes. Estes processos são analisados para então serem elaboradas oportunidades de melhoria.

Dada a competitividade do negócio, a exigência evolutiva do mercado e todas as preocupações regulatórias inerentes, os processos para promover a criação de valor através de software com qualidade para a indústria de apostas necessitam de práticas de engenharia adequadas. Com este objetivo, este projeto foi executado durante seis meses, começando com o uso de ferramentas de modelagem de processos para obter uma visão adequada dos processos analisados. Após ter sido alcançado um forte entendimento de cada processo, o foco passou para a identificação de ineficiências, *bottlenecks* e áreas cinzentas. Aplicando modelos e metodologias de qualidade como *Henry Ford*, *Toyota Production System*, *Total Quality Management* e *Six Sigma*, foi possível identificar a causa principal que motiva cada um dos problemas identificados e a elaboração de soluções para uma seleção de causas.

Por fim, concluiu-se que para melhorar as características funcionais e não funcionais do software produzido é fundamental refinar os processos relacionados com o seu desenvolvimento. Assim, para atuar nas oportunidades de melhoria, foi necessário estudar dados recolhidos para auxiliar na seleção dos *browsers* e sistemas operativos mais valiosos na perspetiva do cliente. Para além disso, foi também necessário trabalhar na reestruturação de processos internos como a documentação de Planos de Teste, um redesenho de todo o processo de libertação de software através da utilização de automação, e a elaboração de documentos baseados em listas para garantir a qualidade do software entregue.

Abstract

The present document was elaborated within the context of the thesis for the completion of the master's degree in Services Engineering and Management of the Faculty of Engineering of the University of Porto. This project aims to analyse a case study related to quality assurance processes inside the betting industry. Thus, for the company under study to gain a steady position on a challenging business from a software engineering perspective, namely the online betting software, the company developed a new application. Hence, this thesis presents a case study in which the processes of feature development and release to the customers are analysed to be further improved.

Given the competitiveness of the business, the exigency of the market and all the regulatory concerns involved, the processes that are undertaken to successfully promote the value creation of software with quality inside the betting industry require suitable engineering practices. Consequently, this project was executed for six months and started by using process modelling tools to get a suitable overview of the processes analysed. Subsequently achieving a strong understanding of each process, the focus was to identify the inefficiencies, bottlenecks and grey areas. By applying manufacturing quality models and methodologies such as Henry Ford, Toyota Production System, Total Quality Management and Six Sigma, it was possible to identify the root cause of each problem discovered and elaborate valuable solutions for a selection of root causes.

Finally, it was concluded that in order to improve the functional and non-functional characteristics of the software produced, it is crucial to improve the processes related to its development. Thus, to act on the opportunities for improvement, it was necessary to perform data analysis studies to aid in the selection of the most valuable browsers and operating systems from a customer perspective. Furthermore, it was also required to work on the restructuring of internal processes such as the documentation of Test Plans, a redesign of the whole release process through usage of automation and the elaboration of checklist-based documents to ensure the quality of the software delivered.

Acknowledgements

I want to express my most sincere gratitude to my coordinating Professor, Professor António Carvalho Brito, for his relentless guidance and effort on supporting the development of this project.

I'm also fortunate that I was able to complete another journey of my academic path at Blip. Thank you to Blip and João Ribeiro for allowing me to produce this work as I most desired to, with the liberty and creativity you got me used to. Thank you for trusting and supporting me while allowing me to take the time I needed throughout this process.

I must also express my gratitude to all my Blip colleagues for your patience over the last couple of months and for your active engagement in the success of this work. The completion of this project could not have been accomplished without your contribution, ideas and experience.

I have no words to describe how much I thank you, João M., for all the advices, encouragement and availability when I most felt like failing. My deepest gratitude for your patient to bear with me for the last couple of months and for your capability to always find the right words.

Finally, thank you, Mom, for being a reference to me. You're the most hard-working person I'll ever meet in all my life and if it wasn't for you, I wouldn't be where I am at such young age. You're my example of how to work through life.

Table of Contents

1 INTRODUCTION 1

1.1 CONTEXT 1

1.2 PROBLEM DESCRIPTION 2

1.3 STUDY AND PROJECT DEVELOPMENT AT BLIP / BETFAIR 3

1.4 RESEARCH QUESTIONS 3

1.5 REPORT OUTLINE 3

2 THEORETICAL BACKGROUND 5

2.1 WHAT IS QUALITY? 5

2.2 QUALITY IN THE MANUFACTURING SECTOR 5

2.2.1 Henry Ford 6

2.2.2 Toyota Production System 7

2.2.3 Total Quality Management 8

2.2.4 Six Sigma 9

2.3 QUALITY CONCERNS EXPANSION TO SOFTWARE DEVELOPMENT 9

2.3.1 ISO/IEC 25010:2011 10

2.3.2 Capability Maturity Model 12

2.4 RESEARCH GAP 13

3 CASE STUDY CHARACTERIZATION 15

3.1 COMPANY BUSINESS MODEL 15

3.2 AS-IS PROCESS 16

3.2.1 Feature Development Process 17

3.2.2 Release Process 18

4 METHODOLOGY 21

4.1 MEASURE PHASE 21

4.2 ANALYSE PHASE 23

4.3 IMPROVE PHASE 23

4.4 CONTROL PHASE 24

5 RESULTS 25

5.1 DATA COLLECTION AND ANALYSIS 25

5.2 SOLUTIONS BRAINSTORMED 26

5.3 SOLUTIONS DEVELOPED 28
5.3.1 Data analysis to select support for browsers and devices - S5 ..... *28*
5.3.2 Analysis of current usage of Test Plans documentation and redefinition – S6 ..... *33*
5.3.3 Release Process Re-design – S10 ..... *36*
5.3.4 Checklist document comprising the most critical validations – S11 ..... *37*
6 CONCLUSION AND FUTURE RESEARCH 39
6.1 MAIN CONCLUSIONS 39
6.2 FUTURE RESEARCH 41
REFERENCES 42
APPENDIX A: ISO 25010 STANDARDS 45
APPENDIX B: NON-EXHAUSTIVE EXAMPLE OF BETTING OVERVIEW 47
APPENDIX C: FEATURE DEVELOPMENT PROCESS MAP 50
APPENDIX D: EXAMPLE OF A RELEASE SCOPE 51
APPENDIX E: RELEASE PROCESS MAP 52
APPENDIX F: PROBLEMS AND ROOT CAUSES ANALYSIS FOR THE FEATURE DEVELOPMENT PHASE 53
APPENDIX G: PROBLEMS AND ROOT CAUSES ANALYSIS FOR THE RELEASE PROCESS . 57
APPENDIX H: RAW DATA OF BROWSERS USED 58
APPENDIX I: ANDROID AND IOS USAGE JANUARY-JULY 2020 AND VARIATION 2020-2021 59
APPENDIX J: DEVICES PER QA 60
APPENDIX K: RELEASE PROCESS MAP AFTER CHANGES 61
APPENDIX L: SEVERITY CLASSIFICATION AND PRIORITY 62
APPENDIX M: RELEASE PROCESS CHECKLIST 63

List of Tables

TABLE 1 - ISO/IEC 25010 STANDARDS 11
TABLE 2 - DMAIC PHASES 21
TABLE 3 - PROBLEMS AND ROOT CAUSES OF THE FEATURE DEVELOPMENT PROCESS 25
TABLE 4 - PROBLEMS AND ROOT CAUSES OF THE RELEASE PROCESS 26
TABLE 5 - SUGGESTED SOLUTIONS FOR THE ROOT CAUSES 26
TABLE 6 - BROWSERS PERCENTAGE OF USAGE 29
TABLE 7 - BROWSERS PERCENTAGE OF BETS SETTLED VALUE 30
TABLE 8 - OPERATING SYSTEMS PERCENTAGE OF USAGE 31
TABLE 9 - ANDROID USAGE FROM JANUARY TO JULY 2021 31
TABLE 10 - IOS USAGE FROM JANUARY TO JULY 2021 32
TABLE 11 - GO/NO-GO CRITERIA 38
TABLE 12 - TYPES OF BETS 47
TABLE 13 - TYPES OF BETTING METHODS 47
TABLE 14 - FOOTBALL MARKET TYPES 49
TABLE 15 - HORSE RACING MARKET TYPES 49
TABLE 16 - ANDROID USAGE FROM JANUARY 2020 TO JULY 2020 59
TABLE 17 - IOS USAGE FROM JANUARY 2020 TO JULY 2020 59
TABLE 18 - IOS DEVICES PER QA 60
TABLE 19 - ANDROID DEVICES PER QA 60

List of Figures

FIGURE 1 - KEY PROCESS AREAS OF CMM QUALITY MODEL 13
FIGURE 2 - IMPACT-EFFORT ANALYSIS OF EACH SOLUTION 28
FIGURE 3 - RELATIONSHIP CODEBASE AND TEST PLAN 35
FIGURE 4 - RELATIONSHIP CODEBASE AND USER STORY 36
FIGURE 5 - EXAMPLE OF A SPORTSBOOK MARKET 47
FIGURE 6 - EXAMPLE OF AN EXCHANGE MARKET 48

List of abbreviations

API	Application Programming Interface
CMM	Capability Maturity Model
DMADV	Define Measure Analyse Design Verify
DMAIC	Define Measure Analyse Improve Control
DOR	Definition of Ready
FEUP	Faculty of Engineering of the University of Porto
GDPR	General Data Protection Regulation
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization of Standardization
KPA	Key Process Areas
MESG	Master in Service Engineering and Management
OS	Operating System
QA	Quality Assurance
QC	Quality Control
QM	Quality Management
RCA	Root Cause Analysis
TPS	Toyota Production System
TQM	Total Quality Management
US	User Story

1 Introduction

The present thesis was elaborated within the scope of the Master in Services Engineering and Management (MESG) of the Faculty of Engineering of the University of Porto (FEUP).

The document focus on the analysis and improvement of quality assurance (QA) processes within the software development area for the online betting industry.

In this section, the goal is to provide a general overview of the context of this study as well as the research objectives and document organization.

1.1 Context

According to the European Gaming and Betting Association (EGBA, 2019), the technological innovation is allowing the European online gambling market – composed of Sports Betting, Casino, Poker, Bingo and Lottery - to grow about 10% per year with the endnote of having more and more customers using online platforms rather than the land-based solutions as local casinos or bookmarker shops. While achieving the steady high growth rates over the past years, the coronavirus pandemic brought a significant drop in revenue during Q2 of 2020 due to the cancellation of all major European sports events. Despite this dip, the remaining quarters of 2020 allowed the market to increase by 7% in revenue, while the European land-based alternatives suffered a drop of 33% in revenue the same year (EGBA, 2020a). Due to the growth horizon and the resilient of the sector during the crisis (Statista, 2020), the growth in the online option is predicted to continue to spread over the years, reaching an estimation of €111bn market worth in 2025 comparing to €97.2bn in 2021 (EGBA, 2020b).

The characteristic of rapidly growth expectations of this market associated with the technology advances places it within one of the most challenging ones from a software engineering perspective (Marinkovic et al., 2020). As a consequence, in order to gain a competitive advantage and a steady place within the market, is a sector that needs regular maintenance and analysis of current client needs and expectations of the service - frequently change-driven by competitors.

Among all online gambling activities, Sports Betting online platforms appear to be the most popular when comparing with the other activities within the European sector (EGBA, 2019). Adding to the challenges present before concerning all sector, Sports Betting is also characterized by having highly rigorous and challenging software requirements, both functional such as local and international regulations, among others, as well as non-functional concerns such as performance and portability (Marinkovic et al., 2020).

The exigency of this competitive-intensive industry requires the processes within the Software Development Lifecycle, especially the Testing and Quality Control, to be well integrated in order to provide a product that works as expected with the desired level of product' quality agreed on. For Montgomery, "a business that can delight customers by improving and controlling quality can dominate its competitors" (Montgomery, 2013). According to the International Software Testing Qualifications Board Foundation Level Syllabus (ISTQB, 2018), "most people have had an experience with software that did not work as expected (…) that can lead to many problems, including loss of money, time, or business reputation, and even injury and death". In this context, the defects would incur the loss of money as well as

causing an impact onto the business reputation which can prove to be substantially harmful to companies within this competitive-intensive industry.

1.2 Problem Description

The software developed for online gambling, in particular Sports Betting, has to provide several order winners to enable the competitive advantage in this rapidly changing and growing market. Apart from the constant maintenance and study of customer trends, there are also highly rigorous requirements and standards the organizations must comply with.

Primarily, the gambling industry is a sector under a lot of regulation due to the large quantities of money being transacted every minute, having several anti-money laundering guidelines to follow. Besides, it's a sector commonly known for its tendency to addictions, thus, the online bookmarkers have to involved in social responsibility initiatives to promote responsible gambling. Considering that every country has its own regulation regarding which types of betting and terms they approve on the region, betting platforms must include tailored considerations for each country they desire to explore. Finally, and in common with every software dealing with sensitive customer data such as addresses, personal identification numbers and credit card details, the software must provide the required level of security.

Regarding functional requirements, apart from the bureaucratic reasoning related to the local and international regulations in part related to Gambling Safety, it's also necessary to provide to the customer several different types of sports on the platforms, highly based on the popularity of the local sport (e.g. providing Baseball in the United States of America and Football in the United Kingdom). Furthermore, the sector must offer a variety of events to bet (see Appendixes A) as well as the provision of the possibility to place bets during in-play games. To achieve so, it's required to have a stable integration with third-party data feeds responsible for the provision of data related to the game stats and incidents that produce variations on the expected likelihood of that event to happen, also known as *odds*. The availability of bets placement during in-play events require the system to achieve different quality factors[1] (See Appendix A for an extensive list), namely performance efficiency which is "the degree to which a component or system uses time, resource and capacity (…)" (ISTQB, 2015) and reliability which implies "the degree to which a component or system performs specified functions under specified conditions for a specified period of time" (ISTQB, 2015). Other than that, to meet the customer expectations more successfully, further non-functional requirements are indispensable. These can be the ability of the software to be transferred from one system to another while still working as expected, also designated as portability (ISTQB, 2015) and mandatory required levels of security.

To build a system with these capabilities, several strict procedures should be carried out in order to ensure quality and compliance with the customer's expectations, organization's standards and legal regulations. Hence, the problem under analysis on this project is the

[1] Also known as non-functional requirements on FURPS quality model and Quality Standards for ISO 25010.

correlation between the internal processes undertaken to ensure the quality of betting software under the company case-studied, and the achievement of external quality attributes.

1.3 Study and Project Development at Blip / Betfair

This project was developed at Blip, a Porto-based software development company focused on the development of entertainment software such as betting platforms. In 2012, the company was acquired by the British group Betfair and in 2020, the former group was reinstructed inside Flutter Entertainment plc[2], a global company that offers Sports Betting, Gaming and Entertainment for over 14 million customers worldwide.

With the increasing importance of the online entertainment industry, Betfair decided to develop a new application. This new application has code-shared and developed by 120 developers and combines not only Sports Betting but also Gaming opportunities such as casino-like entertainment. Customer engagement and customer acquisition teams are also involved which makes this project in fact challenging in terms of both maintenance and assurance of quality.

The presented project was carried for 6 months, from March to August 2021. During the deployment of the project, the initial focus was the analysis of the quality process inside the software development project, followed by the identification of the problems and its root causes to be further possible to develop the most adequate solutions to improve the day-to-day of the people involved on the process as well as the quality of the software produced.

1.4 Research Questions

In order to provide guidance on the elaboration of the present project and considering the objectives of the study, the following research questions were raised with the aim to improve the quality processes inside the organization:

- Research question 1 (RQ 1) – What is the as-is process of quality assurance undertaken in the company?

- Research question 2 (RQ 2) – What problems, inefficiencies and bottlenecks are affecting the company's quality assurance processes?

- Research question 3 (RQ 3) – How to overcome the problems identified?

1.5 Report outline

The present document is divided into six chapters, addressing the following topics: an introduction of the project, the relevant literature for the study, the problem characterization,

[2] Composed of the brands Betfair, Paddy Power, Fanduel, FOX Bet, Sportsbet, TVG, Adjarabet, PokerStars, Sky Betting and Gaming, Oddschecker and Airton Risk Management

the methodology and the data collection chosen, the results discovered through that data as well as the opportunities for improvement found, the proposed solutions and a conclusion.

In the first chapter, an introduction of the project is described to contextualize the main motivations of this project, along with the presentation of the company that serves as a case study. Also, the research questions are raised to be used as a base of the following work developed in this study.

The following chapter comprises a review of the existing literature on the Quality Control area and its theoretical background, starting with an analysis of the primordial of the application of the Quality process to its applicability and importance to the Software Development industry.

The third chapter exhibits the *modus operandi* of the area under study on the company case-studied by providing an extensive description of the process undertaken to create value and the problems underlying in it, to be further analysed.

The fourth chapter presents the methodology chosen to develop the study and its relation to the approaches for Quality presented in the Theoretical Background chapter. This chapter also presents the data collection methodology chosen, as well as a proper justification of the approaches chosen.

The fifth chapter is dedicated to the results achieved through the data collection method chosen in the previous chapter that allows the discovering of the problems faced in the case-studied organization and its root causes. Also, this chapter is devoted to the study, selection and development of suggested solutions to act on the improvement opportunities found.

Finally, the last chapter exhibits the project's findings as well as the opportunities for future work.

2 Theoretical background

The present chapter analyses the relevant literature for this research. The literature review focused on understanding what is quality, and models and management approaches to attain quality inside organizations, composed by a set of key rules and standards. The research was performed on a timeline based, starting with the investigation of quality processes created during the 19th and 20th centuries applied to the manufacturing industry until the start of the digitalization era with the application of models to software development.

2.1 What is quality?

Although terms such as Quality Assurance (QA) and Quality Control (QC) are relatively recent, the work around the area is being deployed since the Middle Ages (Young, 1982). What differs the present from the past is the understanding of the term *quality* as well as the implementation of Quality Management (QM) inside the organizations.

Authors and dictionaries don't appear to reach a consensus on the definition of the term *quality*. To Cambridge dictionary, quality is "how good or bad something is" (2021), and Oxford expands the definition by adding that quality is "the standard of something when it is compared to other things like it" (2021). Some researchers state that producers and handlers tend to classify quality as product-oriented, being associate with specific attributes of the products or services. On the other hand, consumers, marketers and economist are more inclined to a customer-oriented quality assessment, being measured by the achieving of customer desires or necessities with a deep understanding of customer behaviour and tendencies (Shewfelt, 1999). To Montgomery, quality can be defined as the fitness for use that can be evaluated into two different scopes: quality of design and quality of conformance. Quality of design is related to the intentional design differences among products and services that have the same order qualifiers to compete on an equal basis (Montgomery, 2013), and can be defined by materials used in the manufacturing product perspective or characteristic flexibility of the services sector. The term quality of conformance is described as how successful the product achieves the requirements imposed by design that can be influenced by the training of the workforce as well as their motivation, and the processes that control and test the activities being performed. Although this definition is not based on technology and software development experience, the definition is likewise applicable to the scope of this project that will probe more quality of conformance rather than the quality of design.

2.2 Quality in the manufacturing sector

Along with the definition, the introduction and application of quality inside organizations also evolved. The evolution can be explained by the necessity of having more specific technical requirements for the solutions being designed as well as the constantly changing customer's needs and expectations motivated by the vast offer of certain products or services. Those facts require the companies to stand out from the competitors in order to achieve competitive advantage (Christensen, 2001) which is not combinable with products or services with faults that don't allow the fulfilment of the desired needs they were built for, hence, not achieving quality of conformance. This is aligned with the modern definition of quality by Montgomery, which underlines that it is inversely proportional to the variability – related to harmful and unwanted behaviours. Thus, if we are able to decrease the most the occurrence of damaging

events, the quality of a system is improved proportionally. This improvement can be monitored by the achievement of lower costs, processes running smoother and faster, on a generally increased quality perceived by the customer and on the reduction of waste[3] (Montgomery, 2013). The benefits accomplished by introducing concern with quality gained increased importance during the 19th-century Industry Revolution, which promoted the spread of the perspective beyond the manufacturing sector to areas such as services in general such as healthcare, distributors, transportation companies, education and government (Montgomery, 2013).

Nevertheless, the modern concept of quality presented before steps away from the quality implementation of centuries ago that was solely focused on inspection activities with no concern on the customer point-of-view. The first concern with quality that was focused on inspection tenets was based on a post-production review through measuring, examination and testing to identify sources of non-conformance when comparing with company' standards and specification and the definition of a corrective post-action when necessary (Dahlgaard et al., 2008). It was an approach highly focused on the control of the workforce, combining the inspection of the products manufactured with the regular supervision of the employees (Mukhopadhyay, 2020). This approach evolved to Quality Control that extends the activities of inspection to the operational techniques used to reach the requirements for quality (Dale & Plunkett, 1999). These tasks require quality planning and are performed often aided by a limited quality criteria list on a paper-based system combined with self-inspection throughout the production (Mukhopadhyay, 2020). Subsequent, the concept of Quality Assurance arose. This approach combines all the previous with the continuous maintenance of processes that ensures the quality levels through actions of preventions and assurance that the quality issues detected are properly resolved (Montgomery, 2013). It is an approach that relies on the delegation of involvement and cause-effect analysis to successfully increase the company's credibility as well as customer satisfaction (Mukhopadhyay, 2020). Later on, the processes of quality management switched to a broader perspective, focused on the creation of the best processual alternatives to reduce waste with the introduction of concepts such as standardization and just-in-time production as well as a more cultural-wised approach for quality.

2.2.1 Henry Ford

One of the first personalities working with a cultural concept of quality was Henry Ford, founder of Ford Motor Company. By adopting a cultural transformation in its employees' work based on continuous quality improvement and reduced defects and waste, Ford's many industries became world-famous for mass production and economies of scale (Liker, 2004). As a result of his method, the production of a Model T that previously required 12.5 hours, achieved the value of 93 minutes (Georgano, 1985). Ford's tenets that can be applied to the scope of this project – Improvements of Quality process within Software Development - are the following:

[3] Such as money, time and effort (Montgomery, 2013)

- Standardization of processes, tools and equipment (Zarbo & D'Angelo, 2006): "Today's standardization, instead of being a barricade against improvement, is the necessary foundation on which tomorrow's improvement will be based." (Ford, 1926)
- Just-in-time production that reduces stock (Zarbo & D'Angelo, 2006), that could be applied to a backlog of features and technical debt to develop: "Having a stock of raw material or finished goods in excess of requirements is waste…." (Ford, 1926)
- Processes designed that reduce unnecessary moves hence speeding the production or development time;
- Continuous flow production – all the departments are coordinated and when one finishes a task, others can proceed with it the product development;
- Continuous analysis and improvement.

Apart from the reduction of the Ford Model T cycle time, Ford also achieved a significant reduction of the same car cost while increasing the production during the United States of America Depression of 1920. Applied to Software Quality Assurance inside the company case study, the time we take to perform certain tasks due to the associated processes can be analysed and reduced.

2.2.2 Toyota Production System

The Toyota Production System (TPS), created by Taiichi Ohno, is based on the Henry Ford ideology of elimination of waste and just-in-time but also introducing an approach of continuous quality improvement to high-level computerized information systems on a global basis (Zarbo & D'Angelo, 2006). The TPS principles can be described into Philosophy referring to long-term thinking, Processes related to the elimination of waste, People/Partners associated with respect for people and challenge, and Problem Solving focused on a culture of continuous improvement and learning (Liker, 2004). Comparing to Ford, Toyota's market was small due to World War II and presented a portfolio composed of a variety of vehicle's models to satisfy the same market in different ways. This didn't require mass production and economies of scale as on Ford, but operational excellence achieved through the flexibility of operations (Liker, 2004). TPS' tenets that are valuable for the context of the present investigation project are (Liker, 2004):

- Process:
    - Creation of processes to surface problems and allow for a root-cause analysis, such as the "5 Whys" technique;
    - Standardize tasks for continuous improvement and minimization of defects that forces to stop when facing quality problems, applicable to quality concerns inside software improvement;
    - Creation of continuous flow of work combined with visual control so no problems are hidden, applicable trough visual tests of the features developed;

    - Use autonomation but reliable and thoroughly tested technology[4].
- People and Partners:
    - Respect and constantly challenge your people and teams;
    - Empower workforce to grow leaders who live the quality culture and dig for improvement opportunities.
- Problem Solving:
    - Decide slow and implement rapidly: Make decisions by all workforce consensus considering all options.

TPS rules by the constant analysis of workflow through standardized procedures developed to surface problems and correspondent root-cause analysis in order to achieve a learning and improvement culture focused on quality (Zarbo & D'Angelo, 2006).

2.2.3 Total Quality Management

With the evolution and refinement of the area, the approach to quality evolved from the 1920s to the 1980s with the creation of Total Quality Management (TQM). This management perspective can be categorized as the evolution of the concept and application of quality that started from quality inspection, to control, assurance and later on TQM (Dahlgaard et al., 2008). To Montgomery, TQM "is a strategy for implementing and managing quality improvement activities on an organization wide basis".

This approach sums a great cultural change focused on self-responsibility and a culture of quality among all employees - from bottom to top of an organization -, suppliers and customers, also advocating the importance of good relationships between all entities (Mukhopadhyay, 2020). In this perspective, the customer assesses the level of quality determining if the whole organization efforts to foster quality improvement were worthwhile, so the customer focus and involvement are on the priorities. This evolution transformed quality as a management philosophy that puts in place the resources in the most effective way to improve the products and processes combined with cultural change, so every employee embraces the company objectives. TQM involves everyone through teamwork stating that quality is only achieved if sought at all levels of the businesses with an emphasis on quality in design. Thus, every employee is accountable for the success of the product or service being offered (Dahlgaard et al., 2008).

[4] "Toyota, unlike other vehicle manufacturers, doesn't like to try out new technologies. Instead, it prefers proven cutting-edge capabilities that are generating positive results." (12min, 2018)

2.2.4 Six Sigma

In the late 80s, Motorola senior engineer and scientist Bill Smith introduced the method of work Six Sigma that aims to improve the quality of the manufacturing industry by standardizing the way defects are identified and treated in order to attain customer's satisfaction (Smith, 1993).

The Six Sigma is a structured method of work focused on product and service development and strategic process improvement (Linderman et al., 2003). Smith work was developed due to two statement on analysed studies: "A product is built in the shortest time and at the lowest cost if no mistake is made in the process" and "if no defect can be found anywhere in the process of building a product for the customer, then the customer probably will not find one either" (Smith, 1993). Thus, the methodology focus on the reduction of the variability of the outcome of the processes to a level at which defects are unlikely to exist (Montgomery, 2013). Therefore, for each goal task, the method provides structured guidelines on the actions to be performed in order to attain the highlights of the method. These guidelines will be describe further on Chapter 4 of this thesis.

Nowadays, Six Sigma methods are still being used, however integrated with Lean initiatives. The method that started on the manufactory industry spread beyond that to sectors such as the Military, Retail, Technology, among many others, being applied nowadays by companies such as Motorola itself, Cisco, Amazon, between various others that claim the successful application of the concept (6sigma.us, n.d.).

2.3 Quality concerns expansion to software development

With the increasing tendency of digitalization, the approach to quality was also introduced into software development (Miguel et al., 2014). According to the International Organization for Standardization (ISO), software quality is "the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value" (ISO, 2011). Although the term quality has different understandings among authors and organizations, the definition of software quality appears to reach more consensus being that the Institute of Electrical and Electronics Engineers (IEEE) presents a resembling definition. To IEEE, software quality is "1. The degree to which a system, component, or process meets specified requirements. 2. The degree to which a system, component, or process meets customer or user needs or expectations" (IEEE, 1990).

This engineering field evolved from a QC stage only focused on the inspection of the functional side of the components developed, to an additional concern with concepts related to the non-functional architecture that influences the customer quality perception. The rationale behind the expansion of the field is comprehended by the rising technologies, that allow different functional and non-functional capabilities, as well as regulatory requirements imposed by governments or other organizations (Gordieiev et al., 2014), being General Data Protection Regulation (GDPR) an example. Now more than ever, customers are anxious about the overall quality of the software, especially in terms of reliability (Suman et al., 2014), expecting software with zero defects. According to ISTQB, it's a fallacy to expect that the testing activities uncover all existing errors since the goal of testing is to show the presence of defects and not their absence (ISTQB, 2018). ISTQB (2018) also reports that even if a large number of defects is detected and fixed, that's not a guarantee of the success of the product

since it can still not fulfil the users' needs and expectations. These statements place increased importance on all processes involved in the software development, from requirements elicitation, design to coding, testing and deployment. From this perspective, quality is built on an organisational basis, from bottom to top. Notwithstanding, several deviations from requirements and errors are introduced during software development, being this the focus of the current document. If errors are possible to be detected early on, the costs to fix will be lower when comparing to errors find later on in the software development (Suman et al., 2014). One common measurement of software quality is the number of defects found on a released software program, measure as the number of defects per thousand lines of code (Singh & Kannojia, 2013). Some researchers such as Harter and Slaughter state that complex software has significantly lower quality (Harter et al., 2000). It was also verified that under specified time pressure or personal conditions, developers tend to introduce issues that compromise the quality of the software product (Austin, 2001). Further enhanced opportunity to improve quality is based on customer's feedback. Application distributors such as Apple Store or Play Store allow user ratings and reviews such as features requests and bug reports that can be great tools to be used as an opportunity to improve the quality (Genc-Nayebi & Abran, 2017).

Quality gains increased importance in this context that merges the expanding user requirements demanding new functionalities while maintaining the contracted and expected level of non-functional attributes. To achieve these expectations, is expected and needed that the internal processes attain successful results in terms of error-detection, productivity and cost-effectiveness (Fenton & Bieman, 2014).

Fenton (1996) outlines that "there is no guarantee that a 'good' process will lead to a good product, and there is no consensus that the processes mandated in many standards are even 'good'". On the other hand, Pavey (1995) refers that relatively well-defined processes are opportunities for automation, removal of expensive inefficiencies and bottlenecks. Along the same line of reasoning, Arthur (1993) reports that the product quality of software cannot be improved if the processes to developed that software cannot be improved as well. Hence, in resemblance to the approach to quality in the manufacturing industry, numerous quality models have been designed and published over the years. These quality model can be split into the enhancement of software product quality or software process quality, and a few of them concern both (Tate, 2003). For the scope of this thesis, the focus is placed on those that help improve the processes to ensure quality rather than the improving of the software through new functionalities and designs. Notwithstanding, the improvement of processes is influenced by the quality factor it aims to attain, e.g. improvement of processes to validate and verify the portability standards. Thus, is relevant to detail a quality model that focuses on product characteristics such as ISO 25010 combined with a procedural quality model as Capability Maturity Model (CMM) to evaluate the maturity of the processes undertaken to enhance the QM.

2.3.1 ISO/IEC 25010:2011

ISO 25010 is a quality model published by ISO in 2011 as an update of ISO 9126. This model is composed of 8 standard characteristics summarized in Table 1. The goal of this approach is to guide the development and testing of a software product or computer system by ensuring the fulfilling each standard in order to attain quality (Miguel et al., 2014).

Table 1 - ISO/IEC 25010 standards

Quality characteristics	Definition	Quality sub characteristics
Functional Suitability	Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions	Functional Completeness
		Functional Correctness
		Functional Appropriateness
Performance Efficiency	Represents the performance relative to the amount of resources used under stated conditions.	Time Behaviour
		Resource Utilization
		Capacity
Compatibility	Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment.	Co-existence
		Interoperability
Usability	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.	Appropriateness recognizability
		Learnability
		Operability
		User error protection
		User interface aesthetics
		Accessibility
Reliability	Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.	Maturity
		Availability
		Fault tolerance
		Recoverability
Security	Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.	Confidentiality
		Integrity
		Non-repudiation
		Accountability
		Authenticity
Maintainability	Degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements.	Modularity
		Reusability
		Analysability
		Modifiability
		Testability
Portability	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.	Adaptability
		Installability
		Replaceability

Source: (ISO, 2011)

An extensive list of ISO 25010 standards is available on Appendix A.

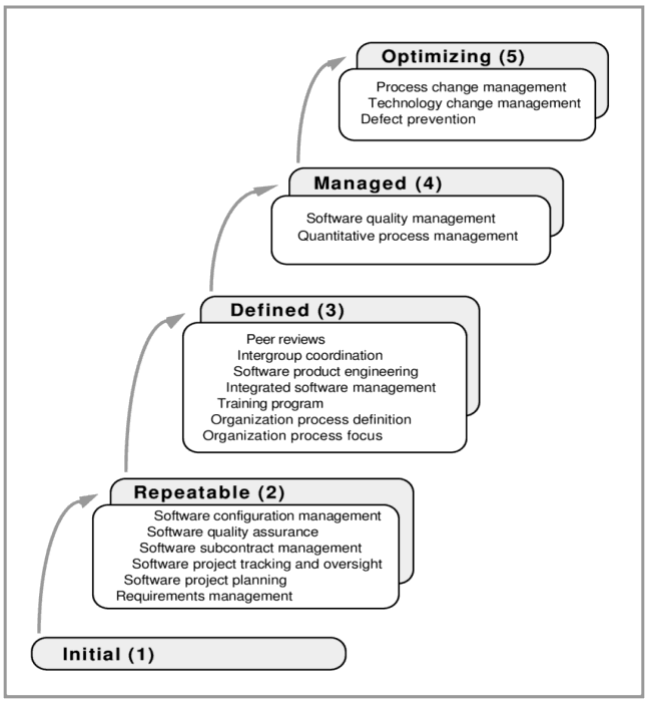
2.3.2 Capability Maturity Model

Capability Maturity Model (CMM) is a process quality model that describes and assess maturity levels of organizations based on the as-is process in place. The objective of the model is to provide guidance to software engineers and managers, in order to select the suited strategies to improve both software quality and ongoing processes (Singh & Kannojia, 2013). As Paulk et al. (1993) underline, by achieving control over processes, organizations can "evolve towards a culture of software engineering and management excellence" with a strong focus on continuous improvement. CMM is organized in five levels of software process maturity and each level specifies the process characteristics accordingly (Jalote, 2000). The five levels are described as the following (Paulk et al., 1993):

1. *Initial*: Software process organization is classified as ad hoc and chaotic. Few processes defined where success depends on individual performance.
2. *Repeatable*: Basic process established to monitor cost, schedule and functionality. Processes that led to success in similar previous applications are put in place. All *Initial* level characteristics are included.
3. *Defined*: Software for the management and engineering activities is documented in a standardized way, as well as integrated with standard software processes. Whole organizations use a tailored version of the standard process of the organization. All *Repeatable* level characteristics are included.
4. *Managed*: Software development processes and products are widely understood and controlled through detailed measures. All *Defined* level characteristics are included.
5. *Optimising*: Collection of quantitative feedback from process and proof-of-concept of innovative ideas and technologies are in place, allowing continuous process improvement. All *Managed* level characteristics are included.

To allow internal process improvement, for each level except the *Initial*, CMM defines key process areas (KPA) with actions that must be addressed to achieve a higher maturity level. The KPA are defined for numerous areas related to software development. However, being

the focus of the present project on the assurance of quality through revision of processes, the detailed KPA relevant for this research are displayed in Figure 1.



Source: (Paulk et al., 1993)

Figure 1 - Key Process Areas of CMM quality model

According to Tate (2003), "the higher the level of the CMM reached, the higher the quality of the organisation's underlying process". Hence, a product with higher quality is most likely to be developed by organizations with a higher level of CMM.

2.4 Research Gap

As presented in detail in the previous sections, the path to quality success within organizations is not only obtained through inspection of the built products or development software, but must be planned, designed, and created through the implementation of a culture of quality. This culture should be spread from bottom to top of a company – including suppliers and customers -, since every member plays a significant role in the delivery of value for the organization. Internally, quality must be considered through every stage of value creation, such as product idealization, planning, design and implementation. Inspection activities namely monitoring and control of the deployment of activities should also be rooted inside organizations.

Several different quality models have been designed, published and applied to various sector such as healthcare, education, transportation, among others, as well as applied to organizations of different sizes; however, the literature review lacks the application of quality management approaches and quality models to the betting online industry, one sector gaining increased market share and importance over the years. Being a critical software that deals with thousands of money transactions per minute, as well as constantly updated information

displayed, there is a latent need to analyse, measure and deploy improved processes to ensure agreement with organization requirements, legislation standards and customer expectations.

Finally, considering this gap, the present document aims to provide an analysis of the betting software quality concerns and processes, through the application of quality models to the organization under study. Additionally, as the software development activity is possible to be projected as a production line of virtual components composed by the steps of the software development lifecycle, it will also be valuable to apply to the scope of this project, the insights of the management quality approaches that raise during the manufacturing era.

3 Case Study Characterization

This chapter begins with a description of the company under study and the context in which it operates, in order to have the required characterization to define the *modus operandi* of the problem's discovery as well as the respective solutions. To achieve so, this chapter provides an overview of the organization, quality processes taken in place and the role of the QA on them.

3.1 Company Business Model

This project was developed in an organizational context, at Blip, a Porto-based software development company. Blip is focused on the development of entertainment software such as betting platforms.

By definition, a safety-critical system is a system whose failure could result in harmful events such as death, injury, illness, and significant damage to the environment or properties (Rierson, 2017). Although not placed on the typical safety-critical systems - such as those used on healthcare, aircrafts or nuclear engineering, among others -, the software produced inside this organization is the main source of income for some of its customers, as well as the overall source of revenue for the company. As ISTQB reports, failures can cause a more drastic outcome such as death and injury, however, can likewise cause loss of money, time or business reputation (ISTQB, 2018). With that being said, the unavailability or degradation of the service being offered disrupts the business, placing itself as a mission-critical software system (Fowler, 2004).

As mentioned on section 1.2 Problem Description, any betting software developed has to comply with several businesses' rules such as tailored information, a variety of different useful features for the users (consult Appendix B for a non-exhaustive overview of features) as well as assuring reliability and the desired levels of performance efficiency. Combined with that, the software must comply with regulatory rules.

With the increasing importance of the online entertainment industry, Betfair began the development of a new product. The company's rationales behind the decision to start this new product were the existence of complementary products (designated Sportsbook and Exchange platforms) that provided poor journeys between them; the customers' satisfaction feedback regarding the software's ease of use was not satisfying; the performance of the existing products was slow when comparing to the competitors.

The new product, named Betfair Rebuild, started being developed in 2019 by 3 developers, 1 QA and 2 architects. On the date of the writing of this project, the new application had code-shared and developed by over 120 employees. In terms of organization, the teams are divided into Sports Betting, Gaming, Customer Engagement and Customer Acquisition. Each delivery team, responsible for the development, is composed of developers, QAs, 1 Manager and 1 Product Owner. The team's general responsibility is to develop a version for a mobile browser application as well as a native application – both for iOS and Android. The QAs are accountable for the testing of each new feature programmed by the developers, and for the overall maintenance and improvement of the project's approach to quality – attained not only through different types of testing but also by having in place the most suitable procedures to

ensure the desired agreement with requirements. Moreover, QAs are also responsible for the release process – deployment of a new version of every application to Production which is an environment available for clients.

Although the project started in 2019, 2020 and 2021 have been the years in which more advancements are being made on the applications, where several important decisions were conceived. Nevertheless, since the application is still in an initial state and not yet available to all customers, organizational and procedural decisions still need to be analysed and agreed on. All the circumstances that have been stated place challenges to this project in terms of maintenance and assurance of quality. The context relies heavily on proper and thorough testing to promote effective early defect detection and prevention.

3.2 As-is Process

In order to produce its value proposition, the company uses the Scrum software development methodology. The sprints are time-boxed at 2 weeks, in which every team has the commitment to deliver the features agreed on the first day of each sprint. Each feature can be either associated to the development of new work, being called as a User Story (US) or it can be related to fixes on the behaviour of the features already implemented, being referred to as Bugs. During the sprint period, the teams perform a daily stand-up meeting to inform the rest of the team members of the current status of the work being developed as well as share any blockers that could be compromising the commitment. Additionally, up to 2 refinement meetings are scheduled per sprint with the objective to clarify any doubts on the requirement specification of the user stories[5] to be picked on the following sprints, as well as to point the user stories based on the effort to develop. The punctuation is given based on the Fibonacci sequence[6], in which the higher the number, the more effort to develop is required. The user stories refined are then to be used on every Scrum planning session to determine the tasks to be worked on during the sprint. Habitually, on the last day of the sprint, a retrospective meeting is carried in order to analyse what went well and the problems the team or all project faced during the 2 weeks. If necessary, action items can arise from this meeting to improve what comprised the team performance during the 2 weeks sprint time. All the team – manager, Product Owner, Developers and QAs - attend these Scrum ceremonies.

More closely related to the day-to-day work of the role of a QA, two activities raise up: the feature development, from the review of requirements to the testing after a feature is developed; and the release process.

[5] Document used in agile methodologies to specify user requirements. User stories are free from any technical specifications of the solution but instead reflects what the user wants the system to do (Yang et al., 2010).

[6] Sequence of numbers starting at 0 and 1, created by adding the previous two numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, and so on (Mitchell, 2019). Inside the company case-studied, the punctuation doesn't go higher than 21.

3.2.1 Feature Development Process

The feature development must start with the requirements elicitation by the Product Owner through the definition and creation of shared documents with the Acceptance Criteria – User Stories. This document is created on the Target Process.

Subsequently, for the User Story to be assigned to a sprint and thus for the development phase to begin, the User Story has to meet the Definition of Ready (DOR). This state is composed of a combination of tasks, such as: the assessment if the US is written in the correct format and containing the requirements that must be met to be capable of marking the US as completed when done (also referred to as Acceptance Criteria); the US is refined and punctuated; and the team has all the required resources to start the following stage. When these criteria are attained, the US has successfully met the DOR and the US can be assigned to a sprint. At this stage, the developer proceeds with the implementation of the feature into a local branch[7]. While the development of the feature is in progress, the QA has the responsibility to elaborate the Test Plan document, containing each step that will be further verified once the development of the feature is finalized. Thus, whenever the developer finalizes the development stage of the respective US, the QA checkouts to the local branch and performs distinct types and levels of testing previously defined.

According to the ISTQB Glossary and applied to the company under study, the QAs perform validations on a component and integration level for this phase of the feature development. Component testing is focused on the verification of units or modules that are separately testable and the code. The typical defects found for component testing include incorrect functionality and data flow problems.

Concerning the Integration testing, the activity is performed through the validation of the interactions between components or systems. These tests are developed using test objects such as interfaces, Application Programming Interface (API) and microservices.

The objectives for these testing levels include the reduction of risk by building higher confidence in the quality of the components, interfaces and interactions; verify if the functional and non-functional behaviours are accordant with requirements; find defects and prevent their spread to higher levels of software testing (ISTQB, 2018).

Diverse distinct types of testing are applied inside the scope of the previously described test levels. The QAs practice functional testing, described by ISTQB (2015) as “testing conducted to evaluate the compliance of a component or system with functional requirements.”, combined with non-functional testing types, related to the assertation of “the compliance of a component or system with non-functional requirements” (ISTQB, 2015). Change-related testing is also performed, divided into Confirmation testing (or re-testing) and regression testing. Confirmation testing stands for the activities conducted after the fix of found defects, with the purpose of ensuring that those defects don’t occur anymore. Regression testing concerns the verification and validation of previous tested component or systems to ensure that the new feature or fix does not introduce any other defects on unchanged parts of the

[7] Independent line of development, usually a copy of the master/main branch (Atlassian, n.d.).

software (ISTQB, 2015). The last two tests are especially relevant within an agile development lifecycle as Scrum due to frequent changes in the code.

Depending on the scope of the User Story, the test types can be performed through manual activities and/or by automation testing, being regression testing a strong candidate of the last approach.

After testing in the local environment and after the user story has approval given by other developers and/or QAs, the User Story code can be merged to the Master branch – principal line of development – and deployed to a remote environment named QA to be tested. At this stage, the User Story is verified one more time by the QAs, usually on a mobile device, simulating the experience of a real user. The testing is performed on an integration level, combined with System testing that focuses on the behaviour and capabilities of a whole system or product, such as a software application, hardware or operating system (ISTQB, 2018). The previously detailed test types are performed inside the ambit of these test levels.

In the case of the User Story not achieving the required compliance with requirements or in the case of any unthought off defect is found, the PO has the responsibility to choose if the US returns to the development stage or if a bug report is created for the issue to be later on fixed.

Once the User Story has the approval of the QA in this environment, the Product Owner performs the final validation on an Acceptance level. This testing level output is the assessment of the system's readiness to be deployed and used by the end-users. By reaching this phase and after achieving a positive Acceptance testing feedback by the PO, the US' Definition of Done criteria is met meaning that the US is ready to be deployed to Production on the following release moment.

The full feature development process map, answering to RQ1, is designed in Appendix C, concerning the stages and relevant times.

3.2.2 Release Process

The release of a new version to Production is usually performed twice a week, being one QA at a time responsible to perform the process, changing in a rotative mode. The responsible QA will be referred to as Release Master.

This process starts with the selection, by the Release Master, of the gold package within a series of release candidate packages, composed by the developed code on the scope of the respective user stories. In order to be considered release candidates, the packages must comply with certain rules and standards that are measure automatically through a pipeline of validations. This verification checks if the code attains the required level of security assessment, as well as successfully ran and pass all automated regression tests. If the package matches the requirements, it is considered a release candidate to the Production environment.

After the gold package – package to be released – is selected, the Release Master has the task to generate the release scope. The scope is composed of a one-line description of each user story to be deployed to Production and is generated by a bot program developed inside the company. The program intersects information between several APIs such as Target Process,

GitLab and Jenkins in order to provide the scope within the selected package. An example of a release scope is available in Appendix D.

The generation of the release scope is followed by the share of the same scope with the remaining QAs, in order to guarantee that the developments featured on the release scope are ready to be deployed to Production. If all QAs approve the scope it means that the validations are finalized with success on the testing environment (QA environment) and the testing performed didn't show the presence of defects or that the defects found are not critical. On the other hand, there are some situations in which the package is not approved: critical defects are found, developments are not finalized, among other sporadic situations. If that's the case, the deployment can't proceed and the Release Master has to postpone the release until the issues are fixed, which requires the start over of the process with the selection of a new gold package to be released. If no concerns are raised regarding the package, the release process can continue, being followed by the approval of the POs and the DMs. This approval follows the same path as the QAs approval since if any problem or concern is raised, the release has to stop to be analysed and corrected. If not, the release can proceed.

The following moment on this process is the promotion of the package to the next environment, called Dark environment. This environment shares configurations with Production environment, being the closest one to Production but only available internally. The package stays on this environment for an average time of one day where tests can be performed before being promoted to the final environment, Production.

The release to Production environment is then performed through the Blue-Green technique which allows for safer releases, providing "continuous delivery with zero or near-zero downtime" (Yang et al., 2010). The traffic of the customers on this application is divided into two data centres and following the Blue-Green deployment, the goal is to change the traffic of all customers to one data centre – "Blue" - while the deployment is being processed on the one with no customer-traffic – "Green". After ensuring that no problems were found on the "Green" data centre, the release can proceed by the deployment of the version on the "Blue" data centre and rebalancing the traffic again on the two data centres. When achieving this stage, the customers have available the new version of the software.

After the deployment is successfully finalized in Production, the Release Master has to inform on three different Slack channels about the success of the release, as well as change the channels release to the number of the package deployed in Production at the moment. Since the user stories lifecycle are composed of different stages of development and being released stage is one of them, the Release Master has the task to update the state of each deployed user story. To do so, the bot program is used again, performing this update automatically after a request by the Release Master.

To keep track of the release, an entity on the Target Process API must be created. On this entity, the release package number must be added by the Release Master, as well as the scope released. This is followed by the last step on the release process, which is the addition of queries on a Confluence page. These queries are needed for some other internal projects. The full release process workflow to answer RQ1 is presented in Appendix E, concerning the stages and relevant times associated.

3.3 Problem Definition

Since Betfair Rebuild' project is at an early staging of development and inherited procedures in place on other products, the need for a stricter definition of new procedures and the necessity for the analysis and improvement of the current in-place processes arose. Therefore, the key areas presenting bottlenecks and problems that require a study are the Feature Development on a QA perspective and the process of release of a new version of the software to the end-user. The bottlenecks are motivated due to time-costly procedures, absence or mismatched information that appears to be a risk for the overall quality of the product being developed.

Hence, this document aims to evaluate the suitability of the current quality assurance processes in a critical context and improve them accordingly. The analysis and improvement will be made using techniques applied in both the software and manufacturing industry such as TQM, Henry Ford, Six Sigma, among others, not yet applied to the online betting industry.

4 Methodology

Six Sigma is a structured methodology of work focused on product and service development and also on strategic process improvement (Linderman et al., 2003). As mentioned in section 2.2.3 Six Sigma, the method provides structured guidelines on the actions to be performed in order to achieve the reduction of variability associated with the methodology. To attain so, Six Sigma uses a five-step problem-solving approach incorporating the phases Define, Measure, Analyse, Improve and Control (DMAIC), to be applied on the analysis and improvement of processes; and Define, Measure, Analyse, Design and Verify (DMADV) centred on the definition and design of a new product or service (Selvi & Majumdar, 2014). For this project's proposed objectives - to identify and analyse problems to further improve the operations -, the DMAIC approach proves to be the most adequate. The Table 2 details the major activities involved in each phase of the method.

Table 2 - DMAIC Phases

Phase	Description	Activities
Define	Definition of the problem, output to be improved, customers and processes associated	Identification of Business case, problem, goals, scope and roles Creation of Process Mapping of the areas to improve
Measure	Collection of data to establish baseline for improvements	Definition of standards performance Development of data collection plan and further collection of data
Analyse	Analysis of the data to find the root causes of defects	Brainstorm the problem causes by inspecting the data collected Examine the value of each step of the process
Improve	Development and implementation of solutions	Analysis of the problems to identify potential solutions Benefits/Impacts are analysed
Control	Implementation of metrics to control improvements	Focus on possible adjustments to be made to on selected solutions

Source: (Selvi & Majumdar, 2014)

Considering that the problem is already identified and that process maps of the context were elaborated and presented in the previous chapter, the *Defined* phase is not mentioned in this section.

4.1 Measure phase

The ground base for the execution and attainability of this project's objectives starts with the collection of relevant data to act as an auxiliary on the discovery of the existing problems to be further studied.

Therefore, the data structure to be collected can be obtained through quantitative research or a qualitative one, as well as a combination of both (Swanson & Holton, 2009). The quantitative approach is characterized by the measurement of social variables in the form of numerical and statistical data (Teixeira, 2020). It is based on a deductive approach since the theories are tested by collecting data to explain a certain phenomenon using mathematical methods (Sukamolson, 2007). According to Swanson and Holton (2009), quantitative methods are strongly entitled to study large samples by applying generalization from a smaller sample to a broader scope.

On the other hand, the authors refer that qualitative research is essential to reach deeper and detailed understandings about any group or sample at the expense of the factor generalizability. According to Silverman (2020), "Quality research is the type of research that finds out about people's experience" and Merriam (2002) states that "The world (…) is not fixed, single, agreed-upon or measurable phenomenon that is assumed to be in positivist, quantitative approach". Thus, the qualitative approach is characterized by the existence of multiple perspectives and interpretations of the same reality and usually have an exploratory aim (Corbin & Strauss, 2008), aiding the development of theories, strategies or actions (Ritchie et al., 2013).

In order to solve problems in the described context, it's required to understand the different perspectives and concerns of each person involved. Therefore, by analysing the objective and characteristics of the project and having in consideration the final work to be presented in the form of several solutions, qualitative research is considered to be the best solution.

Thus, to support the qualitative analysis research, it's required to proceed with the definition of the data collection method that best suits to gather the appropriate information. According to Paradis et al. (2016), "(…) how the information collected is used and what explanations it can generate are determined by the methodology and analytical approach applied by the researcher". Hence, the authors present several data collection methods that can be applied: surveys composed of open-ended formats, individual interviews, focus groups, observations or textual analysis by applying coding grids for analysis.

Upon analysing the advantages and disadvantages of each data collection method through a discussion meeting with a company's manager, the focus group method was the approach preferred to develop this project. The rationale behind the decision was that the different levels of seniority and backgrounds of the involved people combined with the informal interaction created by this approach, allow for pointing out unexpected aspects by the combination of various points of view. Consequently, the group of people combined allows for a richer outcome than just individual interviews.

The data collected through a group interview was performed on the 10th of May 2021. Being the objectives of this project the analysis and improvement on the quality assurance processes inside the company, the focus group interviewers were a group of 8 QAs that volunteered for the project. The group was composed of employees with different levels of seniority and various backgrounds of career experience. Each interview was performed via Zoom platform and recorded with the consensus of the interviewers. The first phase of interviews counted with one non-structured time-boxed meeting of one hour and a half to highlight problems and bottlenecks on the two areas of study: feature development and release process. The data

gathered was then transcribed from the audio recordings to organized tables present on this document.

4.2 Analyse phase

In this phase, the goal is to identify the root causes of the problems discovered in the previous phase. To do so, the Analyse phase aims to highlight and understand thoroughly the problematics under study, so it could be possible to find solutions.

In order to achieve so, a root cause analysis (RCA) was performed through the usage of the "5 Whys" RCA technique, a technique that traces its roots back to the Toyota Production System (Card, 2017). According to Ohno (1988), "The basis of Toyota's scientific approach is to ask why five times whenever we find a problem… By repeating why five times, the nature of the problem as well as its solution becomes clear.". Thus, the tool consists of successively asking "Why?" the necessary number of times until the root causes arrive.

This exercise was performed on the second phase of group interviews that counted with three recorded reunions of one hour and a half each, performed from 31th of May 2021 to 21th of June 2021 via Zoom platform. The group of interviewers was composed of the same 8 QAs employees present in the *Measure* phase and the object of study at which the question was successively repeated was the data collected in the same previous phase. The data gathered was then transcribed from the audio recordings to organized tables present on this document.

4.3 Improve phase

The purpose of this phase is to elaborate and further implement solutions based on the problematics and RCA performed in the preceding phases.

The development of solutions for the root causes identified was performed through another focus group session, counting with the presence of the same interviewers of the previous phase. At this meeting, all root causes were revisited, and a brainstorming session was performed in order to identify the solutions that positively impact the problems faced inside the betting industry's company under study.

Being conscious of the time constraints to develop the current project, it was agreed that it wouldn't be conceivable to examine and develop solutions for every root cause identified. Hence, all the problems were discussed, prioritized and later selected the ones to be studied more thoroughly in the context of this investigation. The prioritization exercise was based on an impact-effort matrix. This decision-making tool aids in deciding which solutions to implement to solve the root causes identified in the previous phases, providing which solutions seem to be the easiest to implement bringing the most positive effects at the same time. The result of this exercise is a visual grid composed of the assignment of each solution to the most adequate category in terms of time and resources necessary to perform. Thus, each solution can fall into one of the following categories: Quick Wins, composed by the solutions that require less effort to develop combined with high impact; Major Projects, related to solutions that provide high impact but demand a greater amount of effort; Fill ins, composed by suggestions that don't require a significant amount of effort but neither offer extensive

benefits; and Time wasters, regarding the solutions that require a great amount of effort at the time that don't provide valuable impact (GroupMap, n.d.). The prioritization exercise was performed on the 30th of June 2021 and the solutions were implemented from the beginning of July to the 13th of August 2021.

4.4 Control phase

As discovered through experimenting, it's known that some solutions appear to have success when first implemented. However, while the projects grow and change, certain options that once were effective, tend to become obsolete and not satisfying the initial purpose.

With this being said, the group decided to perform meetings once six months to analyse the processes in place, following the methodology used on the elaborating of the present thesis. This proposition intends to identify and analyse the obstacles of the processes in place that jeopardize the quality of the work developed in a more recurrent way than it's currently being made. This approach will aid on the day-by-day of the QAs by not allowing the processes to become obsolete as it tends to happen inside corporates, spreading the culture of quality as described on the quality models analysed.

5 Results

After identifying the *modus operandi* of the company and thus answering to RQ1, in order to answer the last two research questions, this chapter focus on the identification of the problems and bottlenecks of the as-is process, as well as its root causes to be further possible to design and implement solutions.

For each identified bottleneck, a corresponding root cause was discussed and assigned. Moreover, for each root cause, solutions were brainstormed and selected to be developed based on an impact-effort analysis required to be performed motivated by the limited time available to develop the current project.

5.1 Data Collection and analysis

Table 3 exhibits the outcome of the 4 interviews[8] undertaken to collect opportunities to improve the Feature Development inside the project, serving as a starting point for the study. Twenty obstacles were raised during a non-structured brainstorming meeting and the respective root causes emerged through the use of the "5 Whys" technique for the identification of root causes from the problems. Appendix F presents the transcribed interviews on the structure of a table, as well as an extensive explanation concerning every problem raised of the Feature Development Process. Thus, this analysis answers to RQ2 concerning the same process.

Table 3 - Problems and Root Causes of the Feature Development Process

Problem ID	Problem	Root Cause
P1	Difficulty to understand what to test	Lack of product context on certain features
P2	Difficulty to find the information regarding features, user stories and bugs	Absence of an efficiently standardized nomenclature
P3	Existence of duplicated bug reports	Absence of an efficiently standardized nomenclature
P4	Lack of understanding on how to start the app to test the development	Lacunas on the communication between Dev – QA when it comes to testing
P5	Visual requirements outdated	Priority: Figma (new design platform) over Abstract (old design platform)
P6	Visual requirements don't follow agreed standards	Final specs are not inspected and fixed
P7	Visual requirements inconsistencies	Final specs are not inspected and fixed
P8	Arduous communication of visual decisions between designers and tech teams	Absence of an available and defined person to answer questions per feature
P9	Issues on the infrastructure that compromise development work	Lack of defined ownership of the infrastructure
P10	Increasing technical debt	Low priority on technical work
P11	Time-costly testing of features	Absence of a rigorously defined list of supported mobile browsers and devices
P12	Deficient management of interdependencies between teams	PO management of expectations: Working on dependencies between teams with close deadlines

[8] One meeting performed to raise problems combined with three sessions to identify the root causes of each problem.

P13	Entropy generated by having 2 different teams working on the same feature for different platforms	Project constantly changing not allowing for strict planning
P14	Constantly changing priorities that compromise quality of native application	Low priority on technical work
P15	UX not considered during development	Rush to start the development of the native application
P16	Entropy created by writing and keeping updated test plans	Missing adequate effort-impact analysis on the action of documenting test plans
P17	Lack of well simulated testing conditions	Testing strategy doesn't include production-like conditions
P18	Difficulty to conciliate QA work streams	Low priority on technical work
P19	Broken web QA environment that jeopardizes QA and local testing	Web: Lack of using of all available tools to avoid breaking environments
P20	It's frequent to have the internal native applications not working for days	Native: Lack of visibility of native failures upon merge to master

Table 4 represents the same analysis for the Release Process that resulted in three opportunities for improvement, thus answering the RQ2 for the Feature Development. Appendix G exhibits a detailed version related to each problem raised and the process undertaken to reach it.

Table 4 - Problems and Root Causes of the Release Process

Problem ID	Problem	Root Cause
P21	Time consuming manual processes	Existing automation tools not fully taken advantage of Lack of a Release Process suitable for Betfair Rebuild particularities
P22	Time-costly last testing validations for the release	Absence of a rigorously defined list of supported mobile browsers and devices
P23	Non-rigorous testing for the releases	Absence of a document comprising the most critical validations

5.2 Solutions Brainstormed

After having identified the root causes for each problem raised and in order to answer RQ3, a set of proposed solutions to solve the issues was brainstormed. For each root cause and having in consideration the problem underlying behind, suggestions were raised to improve both Feature Development and Release Process on a QA perspective, presented in Table 5.

Table 5 - Suggested solutions for the root causes

Problem ID	Solution ID	Suggested Solutions	Root Cause
P4	**S1**	"How to test" section on the Merge Request body for features that require additional steps (such as Prismic configurations or store access)	Lacunas on the communication between Dev – QA when it comes to testing

P8	**S2**	Short-term: Each development team to have a defined point of contact within UX/UI teams Long-term: Low capacity to be solved by hiring more designers	Arduous communication of visual decisions between designers and tech teams
P9	**S3**	Creation of a team dedicated to work on the project's infrastructure Inductions and documents concerning troubleshooting for the most common issues	Lack of defined ownership of the infrastructure
P10, P14, P18	**S4**	Creation of a single technical backlog split by features with a compromise of a determine number of technical User Stories to be picked by sprint	Low priority on technical work
P11, P22	**S5**	Data analysis to select browsers and devices that create more value to Betfair Rebuild project	Absence of a rigorously defined list of supported mobile browsers and devices
P16	**S6**	Analyse current usage and redefine process of documenting Test Plans	Missing adequate effort-impact analysis on the action of documenting Test Plans
P17	**S7**	Automated tests running on real physical devices and on supported browsers	Testing strategy doesn't include production-like conditions
P19	**S8**	Document with all available tools described	Web: Lack of using of all available tools to avoid breaking environments
P20	**S9**	Development of end-to-end native tests	Lack of visibility of native failures upon merge to master
P21	**S10**	Re-design release process	Existing automation tools not fully taken advantage of Lack of a Release Process suitable for Betfair Rebuild particularities
P23	**S11**	Check-list based document with most critical validations to perform	Absence of a document comprising the most critical validations

By preparing and analysing the data collected concerning the identification of the determinants that may be part of the problem, it was identified that certain problems manifest root causes in common with the others, being P11 and P22 examples of that circumstance, among others, as demonstrated in Table 5.

Given the strict timeline to develop the current project and the available internal resources, such as employee's availability and physical resources, not all suggested solutions were possible to be developed. Apart from the time constraints, certain problems raised in the meetings with the QAs do not concern only the QA and the remaining development team but also the Product and the Designers areas. Hence, those problems were selected to not be detailed any further on the scope of this study due to the impossibility to solve with the available resources. With that being said, the removed problems were P1, P2, P3, P5, P6, P7, P12, P13 and P15. Nevertheless, the problems dismissed were shared internally and are now assigned to the respective areas that they concern.

5.3 Solutions Developed

In order to evaluate the most valuable solutions to implement, the effort-impact tool was utilized, as described in Chapter 4.3. The output of this analysis was the visual representation exhibited in Figure 2.

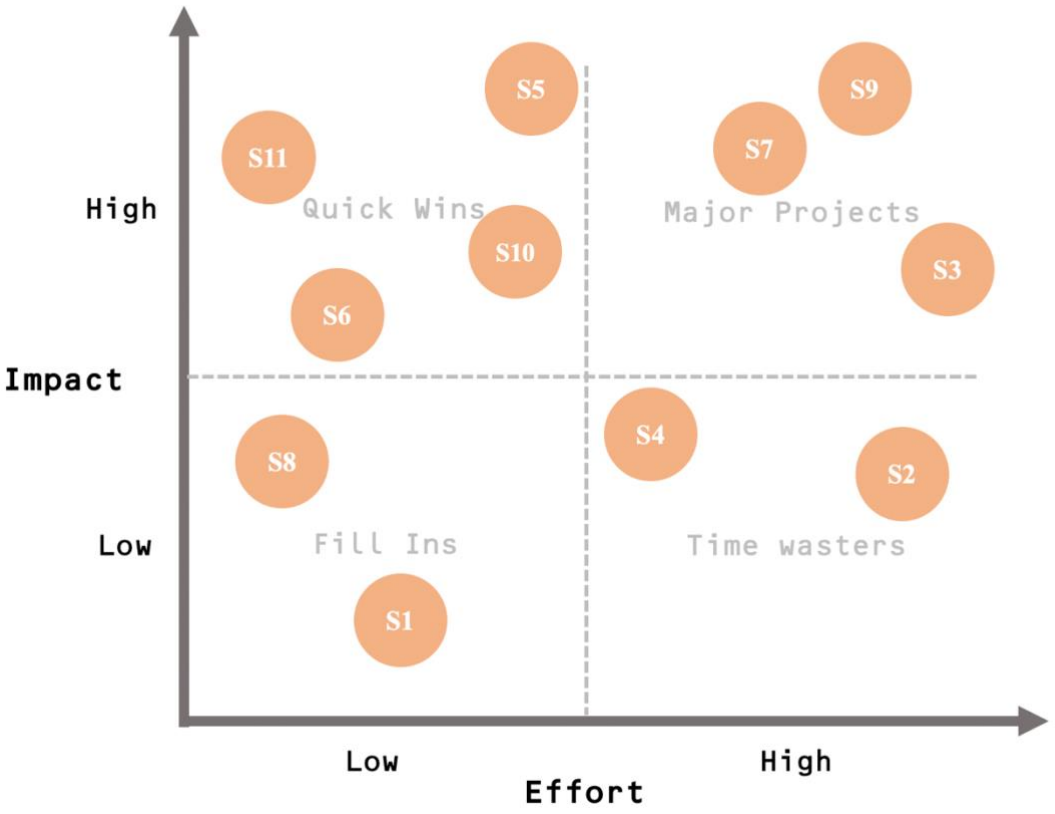


Figure 2 - Impact-Effort Analysis of each solution

Considering the impact-effort analysis performed, the solutions selected to be developed are the ones assigned to the Quick Wins: S5, S6, S10 and S11.

5.3.1 Data analysis to select support for browsers and devices - S5

In order to perform an adequate and informed definition of the browsers and devices support strategy, it is required to have analytical data to support the decisions. The data analysis is valuable and required since it helps on the standardization of procedures, that according to TPS, minimize the occurrence of events that would force the development to stop (Liker, 2004), such as problems on versions or browsers with low value for the business. Since the product is still in a development stage and thus not available to a significant cohort of customers, the data analysed is a combination of information from the two platforms – Betfair Exchange and Betfair Sportsbook - that will be replaced by the new one, Betfair Rebuild. The information was collected through Google Analytics, a tool to track website's activity such as user's accesses, user's localizations, average engagement time and audience's browsers and devices information, particularly relevant for the context of this study.

According to internal documents that exhibit the decisions and rationales agreed on at the commencement of the Betfair Rebuild project, there are seven supported browsers: Google Chrome, Samsung Internet, Safari, Firefox, Opera Mini, Opera and UC Browser. Given the extensive list and according to ISTQB principle referring that exhaustive testing is impossible (ISTQB, 2018), it's necessary to look at the data in order to ensure the supported browsers are the ones that bring more value to Betfair Rebuild – such in terms of percentage of customers as well as the amount of revenue generated on the browsers - and dismiss those that bring less value by having a high cost to test and fix defects combined with a low percentage of users and income.

After an analysis and data preparation of the raw data collected on Google Analytics (see Appendix H for an example), the information obtained regarding the number of accounts per browser is exhibited in Table 6. According to these statements, the most used browsers by Betfair customers are Google Chrome, Safari and Samsung Internet, together achieving approximaditly 98% of the total number of customers. Due to data privacy concerns, the actual number of accounts is not displayed.

Table 6 - Browsers percentage of usage

Browser	% on total number of accounts
Google Chrome	**68.29%**
Safari	**21.14%**
Samsung Internet	**8.30%**
Amazon Silk	1.20%
Firefox	0.51%
Opera	0.39%
Edge	0.04%
UC Browser	0.04%
Yandex Browser	0.03%
Internet Explorer	0.01%
Maxthon	0.01%
Not Defined	0.04%
Total	**100%**

The analysis of the product's revenue per customer was developed through the inspection of the number of pounds settled on bets per browser. This analysis resulted on the values presented in Table 7. Due to data privacy concerns, the actual value is not displayed. As demonstrate in Table 7, three browsers combine approximately 99% of all bets settled by the customers: Google Chrome, Safari and Samsung Internet.

Table 7 - Browsers percentage of bets settled value

Browser	% on £ settled	Notes
Google Chrome	**49.90%**	-
Safari	**36.19%**	-
Samsung Internet	**12.57%**	-
Amazon Silk	0.47%	Only available for Fire Tablets, Fire TVs and Echo Show devices[9]
Firefox	0.46%	-
Opera	0.38%	-
UC Browser	0.02%	-
Yandex Browser	0.01%	-
Edge	0.00%	-
Opera Mini	0.00%	-
Total	**100%**	-

Given the data presented in Tables 6 and 7 and by crossing the information, it is possible to understand that the browsers with more adoption by the customers tend to be the ones that display the higher amount of funds settled on bets. Furthermore, through an evaluation of the impact versus effort to solve the existent issues on the less adopted browsers, as well as the time compromise promoted by the testing activity (presented in Appendix C) on the original settled list of browsers, it was agreed that the support of 98% of the most used browsers combined with 99% of all the amount settled would be an acceptable settlement. This decision dismisses four browsers from the original list: Firefox, Opera, Opera Mini and UC Browser.

After reducing the list of supported browsers, it's necessary to analyse data to support the determination of which Operating System (OS) and OS versions to support. Table 8 exhibits the OS usage in descending order.

Based on the information presented previously, it was decided that the OS to support are Android and iOS, comprising 99% of all the OS used by Betfair's customers. Nevertheless, each OS has available different versions of its users. However, the more OS versions supported, the greater amount of development and testing time will be required to either deliver features, find and resolve defects. Thus, each different version of OS should be analysed in order to select the desired range to offer support.

[9] According to the information available on Amazon Web Services (AWS) website. Source: (Services, n.d.)

Table 8 - Operating Systems percentage of usage

Operating System	% on total number of accounts
Android	**75.09%**
iOS	**23.89%**
Linux	0.58%
Windows	0.28%
Macintosh	0.09%
BlackBerry	0.04%
Windows Phone	0.03%
Non-relevant OS	0.00%
Total	**100%**

Firstly, it's necessary to evaluate the OS usage tendency developed by Google, Android. According to the raw data available on Google Analytics, the Android usage from January 2021 to July 2021 is presented in Table 9 in descending percentage order.

Table 9 - Android Usage from January to July 2021

Android Version	% on total number of accounts
10	**40.24%**
9	**20.17%**
8	**12.26%**
11	**8.76%**
7	5.53%
6	3.47%
5	1.99%
Remaining Android Versions	1.58%
Total	**100%**

The data from 2021 demonstrates that the most used Android version is Android 10, followed by 9 and 8. The most recently deployed Android version, the 11th version [10], is the fourth most installed version of Betfair's customers. Thus, the data exhibited confirms that Android users

[10] Released on the 8th of September 2020 (Rutnik, 2021)

are not concentrated on the latest versions, which requires the selection of a wide support range.

In order to obtain a decision regarding the Android versions to test and provide support, the evaluation method considered the Androids versions usage worldwide. According to Statista (2021), in June 2021, the 4 most recent Android versions accounted for 88% of all Android users. According to the same source, the achievement of a value close to 100% would only be attained by the support of almost all available Android versions, which alienates from the original problem aiming to be solved. Nevertheless, Betfair Rebuild' Android versions usage follows the same trend, having the 4 most recent Android versions accounting for 86% of all versions available. Considering these data and in accordance with other Betfair' projects, it was accepted to support the 4 most recent versions: Androids 11, 10, 9 and 8, achieving the support of 86% of all Android versions used by Betfair's customers.

In order to understand the behaviour of the clients from one year to the next one to further decided when to dismiss and include new versions, the data of the same period for 2020 was examined – consult Appendix I for more detail. The tendency appears to be the same as in 2021, being the most recently deployed version (Android 10 at the time) the fourth appearing on the list. When comparing 2020 to 2021, it's noticeable that the lower version supported in 2020 suffered a decrease of approximately six percentage points, reducing from 11.65% in 2020 to 5.53% in 2021. On the other hand, the most recent version of 2020, Android 10, exhibits an increasing usage of 33.04 percentage points from 2020 to 2021. According to this information, it was decided that the supported list is updated whenever a new Android version is released by the addition of the same version and removal of the lowest one supported.

After analysing Android support, it's essential to perform the same analysis for the second supported OS, iOS by Apple. According to the raw data available on Google Analytics, the iOS usage from January 2021 to July 2021 is presented in Table 10 in descending percentage order.

Table 10 - iOS Usage from January to July 2021

iOS Version	% on total number of accounts
13	**50.26%**
14	**41,43%**
12	**6.16%**
11	0.90%
10	0.21%
Remaining iOS versions	1.04%
Total	**100%**

The data shows that most accounts use the latest releases of iOS, 14 and 13[11], being the second one the most used. Unlike Android, the most recent iOS versions available tend to be the most used ones.

To proceed with a decision regarding the iOS versions to test and provide support, the evaluation method followed the same approach as in Android: analysis of the iOS tendency usage worldwide. As reported by Statista (2021b) in June 2021, the most recently deployed version - iOS 14 -, accounted alone for 78% of all the iOS users worldwide. However, this statement is contradictory to the data available considering Betfair Rebuild' internal usage percentage, in which iOS 14 is corresponding to 41.43% of the users. The combination of iOS 14 and 13 percentages – according to Betfair' Rebuild data - comprises 92% of all iOS users. Although including a wider percentage than Android, it was agreed that the effort to support another iOS version is attainable, and since iOS 12 accounts for a considerable cohort of revenue combined with a wide number of customers, the supported and tested versions agreed on are iOS 14, 13 and 12, comprising 98% of all iOS users.

In order to determine when to dismiss and include new versions, the data of the same period for 2020 was examined, likewise performed in Android – consult Appendix I for more detail. iOS tendency in 2020 is the same as shown in 2021, being the most recent deployed iOS versions (iOS 13 and 12 at the time) the most used ones. When directly comparing 2020 to 2021, it's easily noticeable the most used version of 2020, iOS 12, suffered an extreme decrease in usage, from 48.35% of users in 2020 to only 6.16% in 2021. According to this information, it was decided that the supported list is updated whenever a new iOS version is released by the addition of the same version and removal of the lowest one supported, likewise performed in Android.

After the definition of the supported browsers and devices list present above, in order to increase the coverage of the testing performed considering the devices access limitation due to remote working imposed by the COVID-19 pandemic, the devices owned by Betfair Rebuild were redistributed by each QA. The distribution was preceded by the necessary updates on the devices to comply with the new support list elaborated and detailed previously. To ensure the availability of a wide range of devices comprising the various OS supported, not all devices were updated. The final list of devices in possession by each QA is available on Appendix J, in which the most used device models by the customers were acquired in more quantity.

The impact of this decision allowed for the reduction of the testing time to approximately half of the original value prior to support's list shortening.

5.3.2 Analysis of current usage of Test Plans documentation and redefinition – S6

The Test Plan is a formal document used to define the steps that will be further implemented through automated regression tests. This document is created on the Target Process platform and should act as a mirror of what is currently being tested through automated regression tests

[11] iOS 14 was released on the 16th of September 2020 while iOS 13 was released approximately one year earlier, on the 19th of September 2019

on the codebase. Thus, every Test Plan has a unique identifier that is used in the code for effects of traceability between the codebase and the Test Plans on Target Process.

When a US that implements a new feature is selected to be developed, the Test Plan is elaborated by the QA from zero. However, some USs do not require the creation of a new Test Plan but rather the modification of pre-existing Test Plans, once elaborated on the context of past USs. Apart from that, other non-predictable situations that often occur during an agile sprint also require the modification of Test Plans. These occurrences are often motivated by changes made by the developers or other involved parties, and since the QAs are the only ones responsible for updating the Test Plans that should act as a mirror of the code, a gap is often created. This results in outdated documents that should not act as sources of truth.

During the focus groups, it was noticeable that a part of the process not working as desired was the update of the Test Plans. By analysing the internal documents, it was discovered that most of the Test Plan documents on Target Process do not match the automated validations on the codebase. Thus, since the QAs rely on the information of the previous documented Test Plans in order to elaborate new plans, the tests designed were based on documents that do not correspond to act was actually implemented on the code.

Considering that the project has contributions of over 120 members contributing to its codebase and by analysing the past experiences, a group composed of developers and QAs concluded that a constantly changing project with several unpredictable events can't cope with strict defined rules and procedures. According to Henry Ford (1926), a mechanism to attain quality is the design of processes with reduced unnecessary moves, speeding the production time on a manufacturing world or the development time when applied to software. Thus, it was necessary to understand the usage and the experience of each QA and developer with the Test Plans on Target Process in order to decide on the most suitable approach focusing on decreasing redundant and unessential steps.

This discussion lead to the conclusion that not all QAs find benefits in the documenting of the Test Plans since it's required constant work and attention on it in order to ensure it does not get outdated, misleading the work developed. However, as not all QAs have the knowledge to implement automation regression tests, the Test Plans are a structured method of defining and communicating the Test Plans to the developers, so they are able to further implement them on the codebase. On the other hand, according to the developers, the traceability between the codebase and the Test Plans on Target Process is valuable since it allows to find the correspondent US. The search for the US is necessary when a new development made by the developer creates conflicts with previous developments. This allows for the understanding of what should be the actual behaviour of some components. However, although possible, not all

Test Plans are linked to the correspondent US so the interested parties can't find the Acceptance Criteria that define such behaviour, as demonstrated in Figure 3.

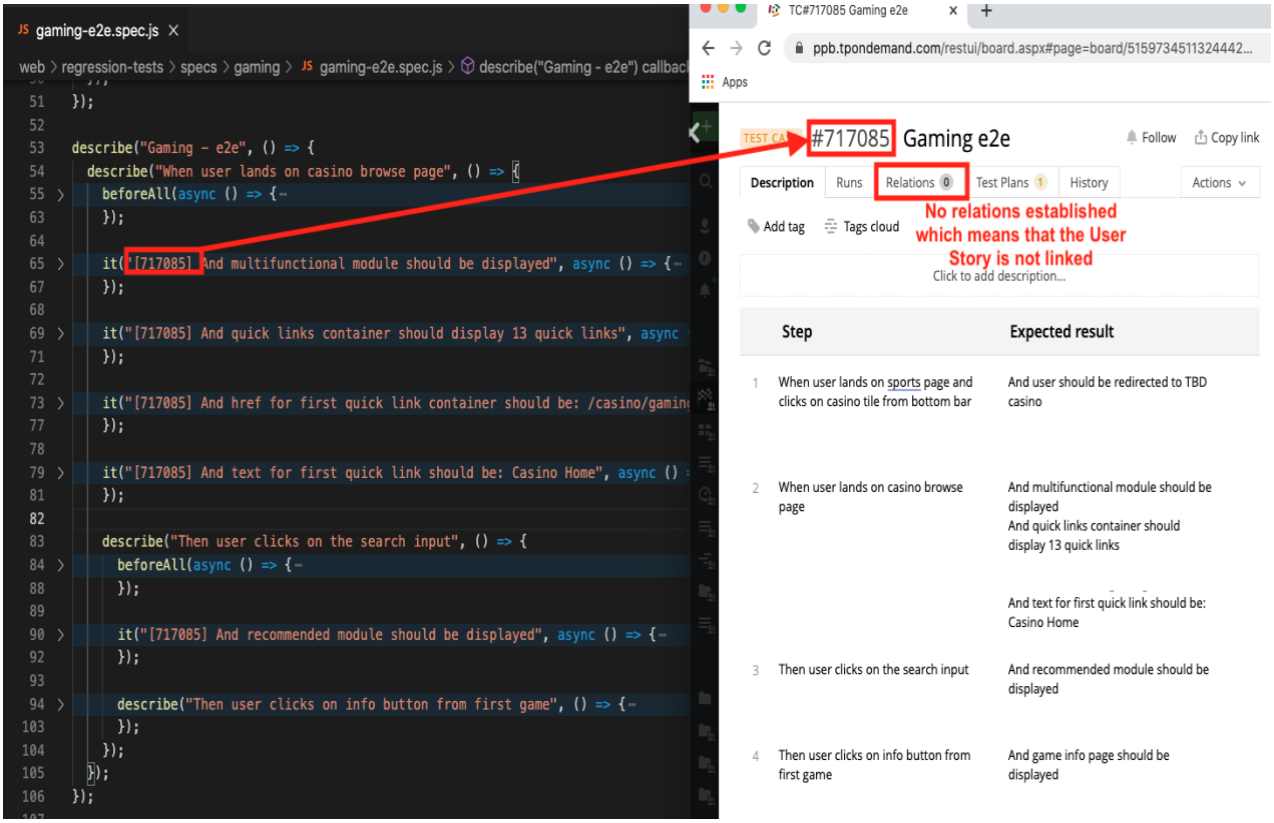


Figure 3 - Relationship Codebase and Test Plan

In some cases, the linkage is correctly performed to a US. However, since the Test Plans are outdated, it means that the corresponding US linked to it is outdated as well.

In order to keep the Test Plans up to date with the tests programmed on the codebase, the option to keep the Test Plans updated after every ran of the automation tests was considered using tools of automation. This solution would minimize the risk of human errors that happen more frequently than desired and thus jeopardize the process. Nevertheless, this improvement suggestion presents to be costly financially and in terms of the time it would take to implement, not delivering significant benefits when compared to the effort required to develop. Furthermore, this alternative doesn't solve the problem of finding and/or linking the US to a Test Plan since it would still be a manual step subjected to human error.

Having these constraints and problems in mind, it was decided that each validation' identifier on the code would not be the identifier of the Test Plan but rather the correspondent US. This allows for a straightforward discovery of the correspondent US required by the developers, as demonstrated in Figure 4.

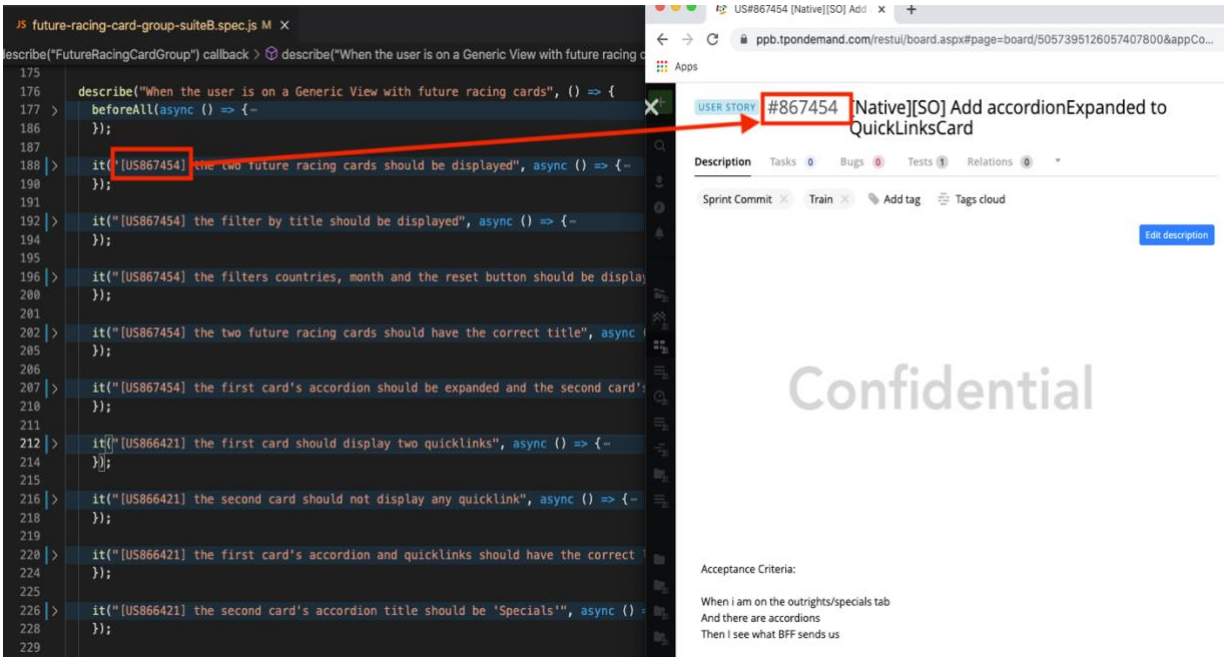


Figure 4 - Relationship Codebase and User Story

Furthermore, this solution proved to solve the remaining problems since it was decided that documenting the Test Plans for effects of communication with other employees would still be made. However, the requirement to have the Test Plans constantly updated would cease to be mandatory. Thus, the source of truth that the QAs now rely on is the tests present on the codebase and not the Test Plans on Target Process, allowing for a more rigorous and less error-prone creation and management of Test Plans. Combined with solving the issues that raised the necessity to refactor this process, this solution also proved to improve the quickness of system recovery when faced with failing tests since it now allows for a prompt discovery of what is the expected behaviour of each development based on the Acceptance Criteria defined on its US.

5.3.3 Release Process Re-design – S10

As described in Chapter 3 and graphically demonstrated in Appendix E, the release of a new version of the application to the client is fairly automated due to the platforms used to aid the activity. Due to been inserted into a rigorous process, the majority of the USs that are deployed to QA environment are ready to be released to the customer, with no requirement of further intensive testing since these validations were previously performed in detail. Nevertheless, after the version is available to the end-customer, the Release Manager has to document the activity, following a bureaucratic set of actions that presents to be time-consuming and often error-prone. Taking advantage of the availability of automation tools, this process was redefined in order to achieve the best use of each tool, as well as the removal of unnecessary steps.

The first action to be analysed was the activity to inform three different Slack channels about the success of the release. Upon analysing, the group concluded that each member was present in at least 2 of the 3 channels where the updates were being sent. Due to the role and

interest of each member of the 3 channels being fairly the same or highly related, the Slack channel with the higher number of members was selected and the stakeholders that weren't a member of the selected channel were added as well. The updates are now only sent on this Slack channel and the remaining channels act as a place for the communication of another release-related matters, as it already happened previously.

The second evaluated action was the update of the channels topic. Each Slack channel has a dedicated area where the channels description can be added. After the release is available for the users, this field was changed by Release Manager to contain the version number of the application available for the customers. However, this step was evaluated and the group concluded that this number does not present any value to the stakeholders on the Slack channel. Hence, this step was removed from the procedure.

The following step that requires manual work by the Release Manager and proved to be error-prone is the creation of an Entity on Target Process to record the release just completed. This document includes the USs deployed to production and the version number of the application available for the client. However highly important, this step does not fully take advantage of the automation tools available inside the project, such as the Release Buddy bot. With this being said, the bot was studied and the group concluded that a feature for the bot to perform the creation of the Entity on its own is possible to be developed. The work to change the bot was not concluded by the time of the writing of this document since the company is working on depreacating internally the Target Process platform in benefit of Jira. Nevertheless, this task is planned to be developed during the third quarter of 2021 already on Jira platform.

The last step that needs rework is the addition of the application queries to a Confluence page. Each application version is composed of different queries that are required in order to allow for internal projects to work. Nevertheless, the projects are used fewer times than the number of queries' updates by the Release Manager after each release. This means that most of the queries added are never used inside the internal projects that require them. With this being said, the group decided to remove this step from the Release Processes and to add it to the troubleshooting document of the respective project that needs the queries.

Although not all changes are yet implemented by the time of the writing of this project, it is predicted that the time needed by the Release Manager to perform the mentioned tasks reduces from 1 hour to 10 minutes. Following these approaches based on the TPS quality model, it was possible to dismiss unnecessary moves that increased significantly the time resources needed to perform the tasks. Furthermore, the use of automation through an intern thoroughly tested and proved reliable technology such as the Release Buddy also significantly aided the process of time reduction.

The refined Release Process is mapped in the Appendix K.

5.3.4 Checklist document comprising the most critical validations – S11

As listed on Appendixes E and L, during the release process, the installation of the Gold Build on different environments is performed: firstly to QA, followed by Dark environment until reaching the destination environment, Production. What differentiates each environment from

the others are the configurations associated with each one. These differences possibility the occurrence of different behaviours by the same Gold Build on the different environments.

In order to ensure that the betting system's ISO/IEC 25010 quality standards such as Compatibility and Adaptability are achieved and maintained, it's expected to have in place mechanisms to ensure coverage of the critical areas of the application. To implement this methodology in Betfair Rebuild' project we must start by the definition of the critical areas, followed by the settlement of rules that evaluate the possibility to proceed with the release when defects are unearthed.

First, what is common between Betfair Rebuild' and all other betting software is its core business: the possibility to place bets. Moreover, to boost its competitive advantage on the market, each platform must provide the most suitable helpers and add-ons for its betting clients: in-play stats, betting tendency graphs, betting promotions like *odds* boosted or bet bonus, among other valuable features for the customer. Furthermore, as with any betting platform, Betfair has to comply with several businesses' rules such as mandatory regulatory requirements settled by organizations specialised in the area. The areas enumerated previously make up the business's critical areas that are required to be included on the release checklist.

In the case of a specific validation failing, it's necessary to evaluate the Go/No-Go criteria of the release. Inside the Betfair Rebuild' project, the decision was taken by the use of a Severity Classification already applied across the company. This classification compromises priorities values from 1 to 5 (P1 to P5), in which P1 is a defect with the highest priority to be fixed while P5 is the lowest. Appendix L presents a detailed explanation of the meaning of each priority level as well as illustrative examples.

For the release process, each validation's criticality was evaluated and associated with a Severity Priority. In the case of the non-successful completion of one or more validations, the Go/No-Go criteria of the release is defined by the rules exhibited in Table 11.

Table 11 - Go/No-Go Criteria

Rules	Go/No-Go Criteria
P1 count >= 1	No-Go
P2 count >= 1	
P3 count >= 1	
P4 count >= 1	Pending Single Evaluation
P5 count >= 1	

Thus, if the checklist entry failing is classified as P1, P2 or P3, the release must not proceed any further and a fix has to be developed in order to finalize that release. On the other hand, if the defect is found on validations classified as P4 and P5, the impact of the failure on the business must be analysed to drive the Go/No-Go decision of the deployment.

The final checklist combining the critical areas elaborated previously and the Severity Classification associated is exhibited in Appendix M.

6 Conclusion and future research

The aim of the development of this project focused on the study and strong understanding of the *modus operandi* of the Quality area inside the company, more precisely the processes of feature development and release to the customers. These processes were graphically modelled, and these maps were crucial in the following stages, such as on the identification of problems, respective root causes and solutions within the opportunities for improvement. To produce results that answer the research questions, management solutions were weighed for each root cause, followed by prioritization motivated by time and resources constraints. The selected solutions were implemented based on adaptation to the company under study to improve the raised problems, and consequently on all the quality of the products delivered.

6.1 Main Conclusions

Given the market competitiveness and strict regulation where the company is inserted, the problems and solutions implemented must help on delivering operational excellence through changing the quality of the end product through improving the processes related to value creation. Thus, through the study performed, it was found that the main problems weren't on the applications themselves but rather on the processes to develop the content and then the procedure related to the release to the customers. The detailing of each step through the usage of process modelling techniques combined with time statements was crucial in providing a global overview of all the steps, the different stakeholders involved in the processes and how they link with each other, thus answering to RQ1.

The availability of process maps modelling the activities was also extremely valuable on the following stage, focused on the analysis of the processes and hence the discovery of problems associated with each one of the two processes under study. Combined with the informal focus groups to gather qualitative data, this exercise allowed for the identification of twenty-three different problems. This approach was valuable in the discovery of problems by highlighting the bottlenecks, the unnecessary steps and the grey areas of each process.

After all the causes have been identified and to analyze and identify the respective root cause, the 5 Why's TPS technique was selected. According to the literature studied, this model was not yet applied to the quality assurance processes in the software sphere. Nevertheless, the exercise proved to be extremely helpful and a tool to be used in the future inside the company in order to identify the root causes of any found defect. Through this study, it was possible to notice that the main problems related to the feature development, and thus answering to RQ2, were provoked by: lacunas on the communication between developers and QAs, absence of defined ownership of projects and infrastructure, the non-existence of a list of supported devices versions and browsers based on customers data analysis, low priority given to technical features, the lack of an adequate effort-impact analysis regarding the documentation of test plans, the lack of usage of all available internal sources and the lack of visibility of native failures upon merging to master. Moreover, the release process root causes identified were related to the automation tools not being fully integrated due to the lack of a suitable procedure applied to Betfair Rebuild' particularities, the non-existence of a list of supported devices versions and browsers and the absence of a checklist-based document comprising the most critical validations to be followed during each release. These causes result in problems such as the absence of critical validations to ensure the quality of the application, the time-

costly testing of features and releases to the customers that makes it difficult for the QAs to conciliate different streams of work, among several other problems described in Table 3 and Table 4.

When comparing the two processes directly, the feature development stands out since twenty problems were raised, while the release process only counts with three problems. This can be explained by the strong usage of automation on the release process that eases some of the tasks and since this procedure doesn't require the participation of so many different members and roles when comparing to the feature development. Apart from that, although the context has changed, the processes inherited the *modus operandi* from older projects developed inside the organization, replicating their same problems.

Motivated by the time and resources constraints that invalidate the possibility to act on the improvement of all the root causes identified, the following stage of this project focused on answering RQ3 through the brainstorming of different solutions followed by a prioritization exercise. For certain causes, it was identified that any improvement ideas would require substantial operational redesign and profound day-to-day changes in wider areas of the company and not only on Betfair Rebuild' delivery teams. Given that the team involved in the project lacked authority on those areas and since the time and operational resources available could not provide significant impact, it was decided to prioritize the work by dismissing certain problems. Nevertheless, these problems were forwarded to the respective areas of concern to be further analyzed thoroughly. For the remaining root causes identified, the group discussed solutions suitable for the *modus operandi* of Betfair Rebuild and the culture of Blip, highly agility-driven with emphasis on a low bureaucratic level.

Due to the limited duration of the project and to provide the desired impact considering the time constraints, after the improvement opportunities identification that was followed by the root causes discovery and understanding, the following phase of the case study was focused on the prioritization of the solutions developed. This exercise was performed by conducting an effort-impact analysis so the solutions that mitigate the problems with a higher impact on the business combined with a relatively low effort to implement were the ones selected to be developed.

Through the application of the solutions, it was clear that the quality of the non-functional, functional and visible characteristics of a software product are strongly impacted by the quality, rigorous and adequacy of the process to develop them. Thus, since the employees are a key part of the successful functioning of any process undertaken, it was required to focus on solutions that sought quality as a management perspective shared at all levels inside the Betfair Rebuild project, inspired by the philosophy of TQM. The emphasis on the need for agility on the steps that the QAs must undertake on the day-to-day provides a stronger recovery system as well as the development of a product that matches its audiences' specific necessities, such as by assuring the quality of the product on the selected browsers, devices and OS versions.

Moreover, the new redesign of processes promoted the reduction of the lead time, highlighting the refactor of part of the release through the removal of unnecessary steps and the better usage of internal automation capabilities, achieving a reduction of 83% of lead time. The usage of process modelling tools not only provided value to have a better understanding of each process by providing a graphic overview but also provided great value in terms of

comparison of the first state of the release process and the redesigned procedure in terms of time, number of tasks and complexity of each step.

The application of quality models designed for the manufacturing sphere proved to be completely adaptable to the context of software development and quality assurance with minor deviations. Hence, this project helped in the elaboration of guidelines to act as a starting point on the identification of opportunities for improvement and further structured analysis to positively refactor any software product, service or process that would require action inside the betting industry.

6.2 Future Research

Given the time constraints and resources limitations motivated, among others, by working remotely during all the extent of this project, certain problems highlighted were not possible to be assessed right away. Hence, after placing aside these improvement opportunities during the last months and within the goal to further improve the quality processes to impact the software products developed, the remaining points will be addressed following the prioritization performed.

Furthermore, the application of the manufacturing-sourced methodologies such as Six Sigma's DMAIC work method combined with statements and tools of other quality models namely Henry Ford, TPS and TQM promoted a strong impact on the processes analysed mainly in terms of reduction of lead time as well as of complexity of the tasks that often lead to error-prone. According to this result, this practice can operate as a foundation for future research and problem solving inside the betting industry and is expected to be exercised once every six months on the quality area of the company's under study. Ultimately, if proved to generate the desired result, the process can be shared and be applied to other areas and departments within the same company.

References

12min. (2018). *The Toyota Way PDF Summary*. https://blog.12min.com/the-toyota-way-pdf/

6sigma.us. (n.d.). *Customer List*. Retrieved June 15, 2021, from https://www.6sigma.us/six-sigma-customer-list/

Arthur, L. J. (1993). *Improving software quality: An insider's guide to TQM New York* (1st ed.). Wiley.

Atlassian. (n.d.). *Git Branch*. Retrieved May 8, 2021, from https://www.atlassian.com/git/tutorials/using-branches

Austin, R. D. (2001). The Effects of Time Pressure on Quality in Software Development: An Agency Model. *Information Systems Research*, *12*(2), 195–207. https://doi.org/10.1287/isre.12.2.195.9699

Card, A. J. (2017). The problem with '5 whys.' *BMJ Quality & Safety*, *26*(8), 671–677. https://doi.org/10.1136/bmjqs-2016-005849

Christensen, C. M. (2001). The Past and Future of Competitive Advantage. *The Challenges of Multilateralism*, 1–5. https://doi.org/10.2307/j.ctvxkn79b.14

Corbin, J., & Strauss, A. (2008). *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, Inc. https://doi.org/10.4135/9781452230153

Dahlgaard, J. J., Khanji, G. K., & Kristensen, K. (2008). *Fundamentals of Total Quality Management*. Routledge. https://doi.org/10.4324/9780203930021

Dale, B. G., & Plunkett, J. J. (1999). *Quality Costing* (3rd ed.). Gower.

EGBA. (2019). *EU Market*. https://www.egba.eu/eu-market/

EGBA. (2020a). *Europe's Gambling Revenues To Drop 23% In 2020 But Online Maintains Growth – New Data*. https://www.egba.eu/news-post/europes-gambling-revenues-to-drop-23-in-2020-but-online-maintains-growth-new-data/

EGBA. (2020b). European Online Gambling Key Figures 2020. *European Gaming & Betting Association*, 1–4.

Fenton, N. (1996). COUNTERPOINT: Do Standards Improve Quality? *Los Alamitos: IEEE Computer Society*. https://doi.org/10.1109/MS.1996.2

Fenton, N., & Bieman, J. (2014). *Software Metrics: A Rigorous and Practical Approach* (3rd ed.). CRC Press.

Ford, H. (1926). *Today and Tomorrow: Commemorative Edition of Ford's 1926 Classic*. Routledge.

Fowler, K. (2004). Mission-critical and safety-critical development. *IEEE Instrumentation and Measurement Magazine*, *7*(4), 52–59. https://doi.org/10.1109/MIM.2004.1383466

Genc-Nayebi, N., & Abran, A. (2017). A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software*, *125*, 207–219. https://doi.org/10.1016/j.jss.2016.11.027

Georgano, G. N. (1985). *Cars: Early and Vintage, 1886–1930*.

Gordieiev, O., Kharchenko, V., Fominykh, N., & Sklyar, V. (2014). Evolution of Software Quality Models in Context of the Standard ISO 25010. *Proceedings of the Ninth International Conference on Dependability and Complex Systems*, 223–232. https://doi.org/10.1007/978-3-319-07013-1_21

GroupMap. (n.d.). *Impact Effort Matrix*. https://www.groupmap.com/portfolio/impact-effort-matrix/

Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development. *International Conference on Information Systems*, *46*(4), 451–466. https://doi.org/10.1287/mnsc.46.4.451.12056

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology. In *Office* (Vols. 12–1990, Issue 1). https://doi.org/https://doi.org/10.1109/IEEESTD.1990.101064

ISO. (2011). *ISO/IEC 25010*. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

ISTQB. (2015). *ISTQB Standard Glossary of Testing Terms 3.01 Foundation Level*.

ISTQB. (2018). *Certified Tester Foundation Level Syllabus*.

Jalote, P. (2000). *CMM in Practice: Processes for Executing Software Projects at Infosys*. Addison-Wesley Professional.

Liker, J. (2004). *The Toyota Way* (Issue 9). McGraw-Hill. https://doi.org/10.1036/0071392319

Linderman, K., Schroeder, R. G., Zaheer, S., & Choo, A. S. (2003). Six Sigma: A goal-theoretic perspective. *Journal of Operations Management*, *21*(2), 193–203. https://doi.org/10.1016/S0272-6963(02)00087-6

Marinkovic, G., Lukic, N., & Medvidovic, N. (2020). Online sports betting through the prism of software engineering. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 1455–1465. https://doi.org/10.1145/3368089.3417049

Merriam, S. B. (2002). Introduction To Qualitative Research. In *Qualitative research in practice: examples for discussion and analysis* (1st ed.). Jossey-Bass.

Miguel, J., Mauricio, D., & Rodríguez, G. (2014). A Review of Software Quality Models for the Evaluation of Software Products. *International Journal of Software Engineering & Applications*, *5*(6), 31–53. https://doi.org/10.5121/ijsea.2014.5603

Mitchell, C. (2019). *Fibonacci Numbers and Lines Definition and Uses*. https://www.investopedia.com/terms/f/fibonaccilines.asp#citation-1

Montgomery, D. C. (2013). *Introduction to Statistical Quality Control* (7th ed.). John Wiley & Sons, Inc.

Mukhopadhyay, M. (2020). *Total Quality Management in Education* (3rd ed.). SAGE Publications India Pvt Ltd.

Ohno, T., & Bodek, N. (1988). *Toyota Production System*. Productivity Press. https://www.taylorfrancis.com/books/9781000056495

Paradis, E., O'Brien, B., Nimmon, L., Bandiera, G., & Martimianakis, M. A. (Tina). (2016). Design: Selection of Data Collection Methods. *Journal of Graduate Medical Education*, *8*(2), 263–264. https://doi.org/10.4300/JGME-D-16-00098.1

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). Capability maturity model, version 1.1. *IEEE Software*, *10*(4), 18–27. https://doi.org/10.1109/52.219617

Pavey, D. (1995). Safer C: Developing software for high-integrity and safety-critical systems. *Software Testing, Verification and Reliability*, *5*(3), 203–204. https://doi.org/10.1002/stvr.4370050306

Rierson, L. (2017). *Developing safety-critical software: a practical guide for aviation software and DO-178C compliance*. CRC Press.

Ritchie, J., Lewis, J., Nicholls, C. M., & Ormston, R. (2013). *Qualitative Research Practice: A Guide for Social Science Students and Researchers*.

Rutnik, M. (2021). *When will your phone get the Android 11 update? (Updated June 24)*. https://www.androidauthority.com/android-11-update-tracker-1155652/

Selvi, K., & Majumdar, R. (2014). Six Sigma - Overview of DMAIC and DMADV. *International Journal of Innovative Science and Modern Engineering*, *2*(5), 16–19.

Services, A. W. (n.d.). *What is Amazon Silk?* Retrieved June 28, 2021, from https://docs.aws.amazon.com/silk/latest/developerguide/what-is-silk.html

Shewfelt, R. L. (1999). What is quality? *Postharvest Biology and Technology*, *15*(3), 197–200. https://doi.org/10.1016/S0925-5214(98)00084-2

Silverman, D. (2020). *Qualitative Research* (5th ed.). SAGE Publications Ltd.

Singh, B., & Kannojia, S. P. (2013). A review on software quality models. *Proceedings - 2013 International Conference on Communication Systems and Network Technologies, CSNT 2013*, 801–806. https://doi.org/10.1109/CSNT.2013.171

Smith, B. (1993). Six-sigma design (quality control). *IEEE Spectrum*, *30*(9), 43–47. https://doi.org/10.1109/6.275174

Statista. (2020). *Coronavirus pandemic causes dip in gambling market*. https://www.statista.com/markets/409/topic/438/gambling/#statistic1

Statista. (2021a). *Mobile Android operating system market share by version worldwide from January 2018 to June 2021*. https://www.statista.com/statistics/921152/mobile-android-version-share-worldwide/

Statista. (2021b). *Mobile Apple iOS operating system market share by version worldwide from January 2018 to April 2021*. https://www.statista.com/statistics/1118925/mobile-apple-ios-version-share-worldwide/

Sukamolson, S. (2007). Fundamentals of quantitative research. *Language Institute*, 20.

Suman, Wadhwa, M., & Rohtak, M. D. U. (2014). A Comparative Study of Software Quality Models. *International Journal of Computer Science and Information Technologies 5.4*, *5*(4), 5634–5638. https://doi.org/10.1109/ICAETR.2014.7012958

Swanson, R. A., & Holton, E. F. (2009). *Research in Organizations: Foundations and Methods in Inquiry*. Berrett-Koehler Publishers.

Tate, J. (2003). *Software Process Quality Models : A comparative evaluation* [University of Durham]. https://www.cs.york.ac.uk/rts/people/jt/docs/msc.pdf

Teixeira, J. G. (2020). Quantitative Research. In *Introduction to Research Project*.

Yang, B., Sailer, A., & Mohindra, A. (2010). Survey and Evaluation of Blue-Green Deployment Techniques in Cloud Native Environments. In *International Conference on Lean Enterprise Software and Systems: Vol. 65 LNBIP*. https://doi.org/10.1007/978-3-642-16416-3_18

Young, H. H. (1982). A brief history of quality assurance and peer review. *Professional Psychology: Research and Practice*, *13*(1), 9–13. https://doi.org/10.1037//0735-7028.13.1.9

Zarbo, R. J., & D'Angelo, R. (2006). Transforming to a Quality Culture. *Pathology Patterns Reviews*, *126*(suppl_1), S21–S29. https://doi.org/10.1309/kvt7nwvpjr73t4k6

APPENDIX A: ISO 25010 standards

Quality characteristics	Quality sub characteristics	Definition
Functional Suitability	Functional Completeness	Degree to which the set of functions covers all the specified tasks and user objectives.
	Functional Correctness	Degree to which a product or system provides the correct results with the needed degree of precision.
	Functional Appropriateness	Degree to which the functions facilitate the accomplishment of specified tasks and objectives.
Performance Efficiency	Time Behaviour	Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
	Resource Utilization	Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
	Capacity	Degree to which the maximum limits of a product or system parameter meet requirements.
Compatibility	Co-existence	Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
	Interoperability	Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
Usability	Appropriateness recognizability	Degree to which users can recognize whether a product or system is appropriate for their needs.
	Learnability	Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
	Operability	Degree to which a product or system has attributes that make it easy to operate and control.
	User error protection	Degree to which a system protects users against making errors.
	User interface aesthetics	Degree to which a user interface enables pleasing and satisfying interaction for the user.
	Accessibility	Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.
Reliability	Maturity	Degree to which a system, product or component meets needs for reliability under normal operation.
	Availability	Degree to which a system, product or component is operational and accessible when required for use.
	Fault tolerance	Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
	Recoverability	Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

Security	Confidentiality	Degree to which a product or system ensures that data are accessible only to those authorized to have access.
	Integrity	Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data
	Non-repudiation	Degree to which actions or events can be proven to have taken place so that the events or actions cannot be repudiated later.
	Accountability	Degree to which the actions of an entity can be traced uniquely to the entity.
	Authenticity	Degree to which the identity of a subject or resource can be proved to be the one claimed.
Maintainability	Modularity	Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
	Reusability	Degree to which an asset can be used in more than one system, or in building other assets.
	Analysability	Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
	Modifiability	Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
	Testability	Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.
Portability	Adaptability	Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
	Installability	Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.
	Replaceability	Degree to which a product can replace another specified software product for the same purpose in the same environment.

Source: (ISO, 2011)

APPENDIX B: Non-exhaustive example of betting overview

Table 12 - Types of bets

Types	Description
Single bet	Placed on a single event.
Multiple	A single bet on multiple outcomes. All selections must win to guarantee return.
…	

Table 13 - Types of betting methods

Betting Methods	Description
Sportsbook bet	Implicit betting against the odd made available from the online bookmarker. According to Figure 5 referent to Porto vs Chelsea game for the UEFA Champions League competition, the customer has 3 different options to bet on. Porto win, Chelsea win or the draw.  **Figure 5 - Example of a Sportsbook market** If the customer places a €2 bet on Porto's win, the return would be €8.40 if the event happened: $$Return = 4.2\ x\ €2 = €8.40$$ In the case of the Porto's victory hasn't succeeded, the customer loses the amount betted (€2 for this particular example) and the value is retained for the online bookmarker.

Exchange bet	Implicit betting against the odd made available from other online betting players. According to Figure 6 referent to Porto vs Chelsea game for the UEFA Champions League competition, the customer has 6 different options to bet on. On the blue side as known as *Back* bets, the customer has the option to bet in favour of events and on the pink side, *Lay* bets, the customer wages against certain result. Based on Figure 6, the customer can bet that Porto wins, Chelsea wins or the draw, as well as gamble that Porto win does not occur, Chelsea win does not occur or that the draw does not occur.



Figure 6 - Example of an Exchange market

If the customer places a €2 bet on the draw, the return would be €7 if the event happened:

$$Return = 3.5\ x\ €2 = €7$$

In the case of the draw hasn't succeeded, the customer loses the amount betted (€2 for this particular example) and the value is transferred for a player that waged the exact opposite result, Lay bet that the draw does not occur and the bookmarker earns a commission on every winning bet.

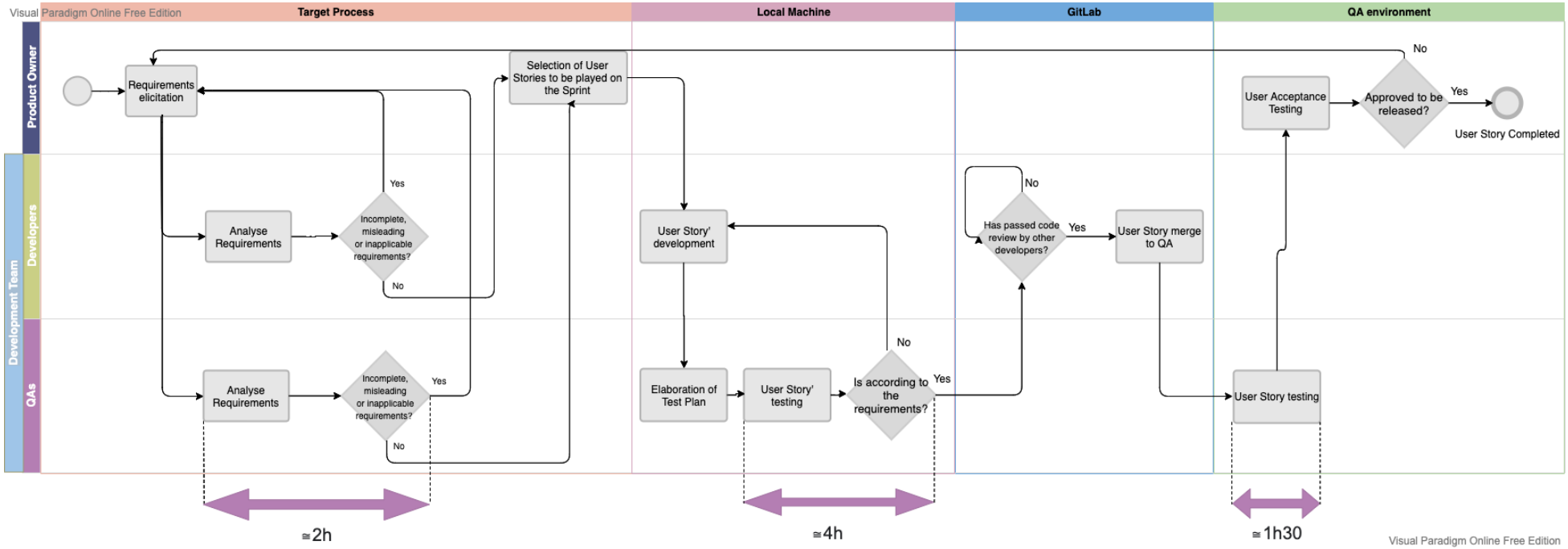
Table 14 - Football market types

Example of Football Market Types	Description
Match Odds / Match Result	Bet on the team that wins the match. Can be either home team, the draw or away team.
Double Chance	Similar to Match Odds but covering two different outcomes.
Over/Under Goals	Bet on the whether there will be more or less goals than a certain number. Under 0.5 Goals means that there are 0 goals on the game and over 0.5 means that there is at least 1 goal.
Both teams to score	Bet on whether both teams score on the match or not.
Correct Score	Bet on the exact correct score of the match.
…	

Table 15 - Horse Racing market types

Example of Horse Racing Market Types	Description
Win	Bet on the horse that wins the match.
Each Way	Bet on the horse that wins the match or stays at least on the 4 or 5 first places (varies according to the race).
Forecast	Bet on the order of the horses that finish in first and second place in a race.
Reverse Forecast	Bet on the first two races finishing the race in no particular order.
…	

APPENDIX C: Feature Development Process Map



APPENDIX D: Example of a release scope

UserStory X - ACCA | My Bets - Integration - **Team**: PT_CTO_Team1 - **Status**: In Progress

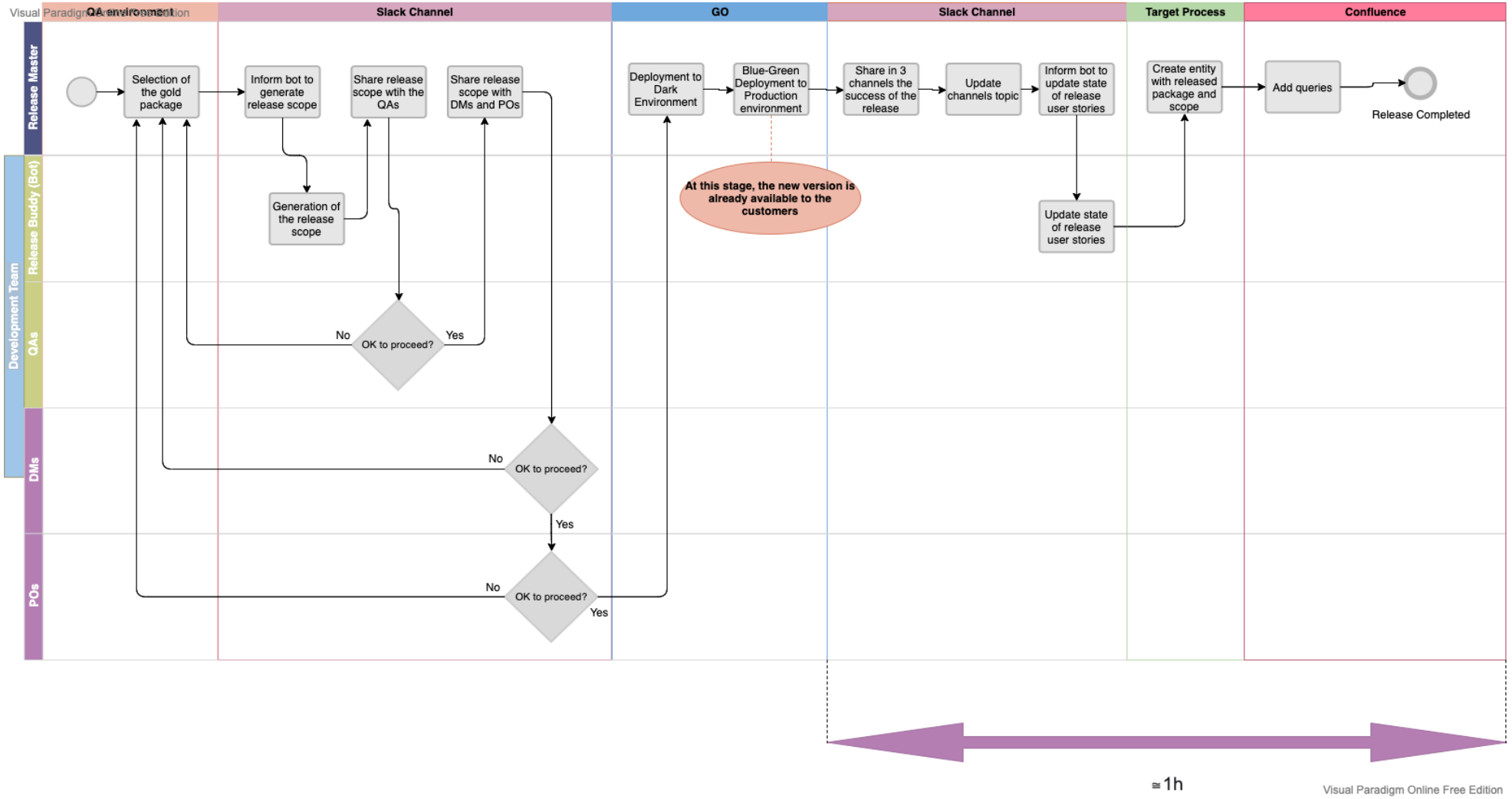
Bug X - CI | GTM - Trigger successful place selection event - **Team**: PT_CTO_ Team1 - **Status**: In Progress

UserStory X - BB | Users can see their Bet Builder Bet on My Bets - **Team**: PT_CTO_ Team1 - **Status**: In Progress

UserStory X - [BFRB][BFF][Regression Test] Live Video - Card query - **Team**: PT_CTO_ Team2 - **Status**: Done

UserStory X - CI | BetBuildersCard Connected Component - **Team**: PT_CTO_ Team2 - **Status**: In Progress

APPENDIX E: Release Process Map



APPENDIX F: Problems and root causes analysis for the feature development phase

Problem ID	Problem	Why?	Root Cause	Explanation
P1	Difficulty to understand what to test	Lack of detailed requirements and acceptance criteria on user stories Lack of knowledge	Lack of product context on certain features	Requirements often based on "implement as it was implemented on the web browser applications" while concerning the native applications requirements; Requirements incomplete including outdated or non-existence visual specifications
P2	Difficulty to find the information regarding features, user stories and bugs	Usage of different nomenclatures for the same functional area or feature Current standardization rules promote this difficulty	Absence of an efficiently standardized nomenclature	While searching for information feature' related, the QAs often face difficulties on efficiently and effectively finding the needed data
P3	Existence of duplicated bug reports	Usage of different nomenclatures for the same functional area or feature	Absence of an efficiently standardized nomenclature	The board that comprises the bug reports has duplicated documents referring to the same bug The board provides a template for the creation of bug reports that doesn't fulfil the needs
P4	Lack of understanding on how to start the app to test the development	Merge Request description presents flaws regarding completeness to aid the QA Gap on the consideration of data for the testing activity	Lacunas on the communication between Dev – QA when it comes to testing	While starting the application, we can point to the dev environment back-end or to a Production one. Certain developments require one type of pointing while others don't.
P5	Visual requirements outdated	Creation of a new design for an older component not followed by the update of older visual specifications Information spread on different design platforms due to migration and lack of capacity to update both	Priority: Figma (new design platform) over Abstract (old design platform)	Existence of different visual specifications for the same component.
P6	Visual requirements don't follow agreed standards		Documentation not shared right from the beginning?	In order to allow for component reusability and to provide a consistent platform, several standards were defined such as: colour pallet, text sizing properties and spacing sizing that should only be multiple of 4 pixels. Still, the visual specifications provided don't follow what was agreed on.

P7	Visual requirements inconsistencies	The extraction from local environment to the Abstract introduces conflicts Final specs are not inspected and fixed	Documentation not shared right from the beginning?	The same component is presented with different properties (colour, sizing, spacing, etc.) on the same visual specification collection
P8	Arduous communication of visual decisions between designers and tech teams	Several different people asking regarding several different features Low capacity of UI/UX team to tackle the questions presented UI/UX team separated from the tech teams	Absence of an available and defined person to answer questions per feature	When faced with doubts, the alignment between tech teams and design teams lingers which delays the implementation
P9	Issues on the infrastructure that compromise development work	Considering that Betfair Rebuild is a project in constant changing composed by more than 100 people, there isn't a general knowledge between all tech members on how to fix all different problems No inductions hosted on how to deal with most common issues	Lack of defined ownership of the infrastructure	The usual day-by-day development is compromised due to environmental issues such as unavailability of Mac Minis (required for the native application) and broken pipelines that don't allow further merges to the common baseline (master branch)
P10	Increasing technical debt	Lack of PO visibility on the necessary documented technical work Lack of understanding on the benefits of required technical work	Low priority on technical work	The technical work raised on the working groups is only a responsibility of the respective group. The work is raised by the groups on a backlog that is not prioritized to be picked during the team's sprints The focus is on feature development and not on technical work
P11	Time-costly testing of features	Supported a wide variety of different mobile browsers and devices	Absence of a rigorously defined list of supported mobile browsers and devices	Due to the large amount of supported mobile browsers and devices, each testing phase takes an average of 4h
P12	Deficient management of interdependencies between teams	Delays on the developments due to infrastructure issues, bugs found, low team capacity,	PO management of expectations: Working on dependencies between teams	Dependencies between teams that happen to clash

		among others that cause delay on dependencies Deficient PO alignment on team's dependencies	with close deadlines	
P13	Entropy generated by having 2 different teams working on the same feature for different platforms	Lack of context on the new feature to de developed Different team implementing for native than the one that implemented for web Team's capacity at the moment of the native feature development	Project constantly changing not allowing for strict planning	Betfair Rebuild is developed in two platforms (mobile web and native application) and sometimes the same feature on both platforms is developed by different teams.
P14	Constantly changing priorities that compromise quality of native application	The focus on feature development rather than technical work such as automation testing	Low priority on technical work	The priority introduced on the native application compromises the quality by not allowing time to work on the project quality.
P15	UX not considered during development	Requirements to develop features as it was developed in web – same requirements and visual specifications	Rush to start the development of the native application	It's often that when presenting the development to designer teams, UX questions and alternatives arise after the development.
P16	Entropy created by writing and keeping updated test plans	Test Plans aren't considered a source of truth because they're outdated The process isn't followed by the whole project	Missing adequate effort-impact analysis on the action of documenting test plans	Every automated test has its test plan on the Target Process platform. Since it's a manual process, it's often that what is being tested on the code is not the same as describe on the test plan.
P17	Lack of well simulated testing conditions	Lack of testing under not favourable conditions (weak wi-fi, memory limitation, etc) Real testing conditions are not documented Lack of testing on real devices and on all supported browsers	Testing strategy doesn't include production-like situations	The tests are currently being performed with devices that are usually turned off. Testing with weak wi-fi, no service, memory limitation due to several apps running in background, between other, is not performed.
P18	Difficulty to conciliate QA work streams	Technical work is not considered as a work stream	Low priority on technical work	Every team works on different features/work streams per sprint. Technical work is not a work stream considered.

P19	Broken web QA environment that jeopardizes QA and local testing	Black screen on the environment that makes it not possible to test in QA	Lack of using of all available tools to avoid breaking environments	Since the tests are performed before and after the merge of the feature to QA, whenever there is a problem with the QA environment, the testing gets delayed.
P20	It's frequent to have the internal native applications not working for days	Pipeline testing strategy doesn’t include real tests	Lack of visibility of native failures upon merge to master	Whenever we merge something on native and the app breaks, merges keep happening above the broken results.

APPENDIX G: Problems and root causes analysis for the release process

Problem ID	Problem	Why?	Root Cause	Explanation
P21	Time consuming manual processes	Several manual actions to be performed Release process emerged from the combination of different release process of other projects	Existing automation tools not fully taken advantage of Lack of a Release Process suitable for Betfair Rebuild particularities	The release process has to follow several processes, being the majority manual processes. See Appendix E.
P23	Time-costly last testing validations for the release	Every person has to test in different devices and on all browsers Supported a wide variety of different mobile browsers and devices	Absence of a rigorously defined list of supported mobile browsers and devices	Due to the large amount of supported mobile browsers and devices, the regression testing phase that anticipates each release takes an average of 2h
P24	Non-rigorous testing for the releases	The validations during the release don't follow a defined checklist Critical areas not defined	Absence of a document comprising the most critical validations	When performing the release to production, the QAs don't follow a strict checklist of validations but rather a non-defined set of tests that can leave aside essential validations

APPENDIX H: Raw data of browsers used

device_type	screen_size	browser	browser_version	os	os_version	accounts
motorola moto g(7)	424x891	chrome	76	android	9	5
motorola moto e5	360x720	chrome	68	android	8	27
leeco lex720 le pro 3	412x732	chrome	78	android	6	1
samsung sm-a600fn galaxy a6	740x360	chrome	77	android	9	20
samsung sm-t580 galaxy tab a 10.1	800x1280	chrome	76	android	7	8
sony f3311 xperia e5	640x360	chrome	76	android	6	10
(not set)	600x1024	chrome	66	android	8	8
lg h502 magna	360x640	chrome	69	android	5	2
at&t k92 primetime	800x1280	chrome	75	android	7	2
oneplus a3003 3	412x732	chrome	68	android	8	4
xiaomi redmi 5	360x720	chrome	81	android	8	22
essential ph-1 phone	438x854	chrome	74	android	9	1
samsung sm-j730f galaxy j7 pro	360x640	chrome	80	android	7	1
sony g8141 xperia xz premium	360x640	chrome	74	android	8	15
wiko lenny3 lenny 3	360x640	chrome	79	android	6	7
samsung sm-t520 galaxy tab pro 10.1	800x1280	chrome	69	android	7	1
alcatel 5033x 1	640x320	chrome	78	android	8	6
sony f5121 xperia x	360x640	chrome	81	android	8	16
lenovo a7010a48 vibe k4 note	360x640	chrome	50	android	6	3
motorola one action	360x840	chrome	81	android	10	54
xiaomi redmi 6a	424x848	chrome	80	android	8	1
zte blade a510	360x640	chrome	75	android	6	3
lg lg-k220 x power	360x640	chrome	75	android	6	75
samsung sm-t819 galaxy tab s2 9.7	768x1024	chrome	78	android	7	12
huawei ane-lx1 p20 lite	360x760	chrome	65	android	8	7
alcatel 5099y 3v	720x360	chrome	80	android	8	4
huawei ane-lx3 p20 lite	360x760	chrome	77	android	9	9
bq aquaris u2	640x360	chrome	79	android	8	5
google nexus 5	773x538	chrome	78	android	6	1
samsung sm-t580 galaxy tab a 10.1	1098x686	chrome	75	android	8	26
(not set)	712x1138	chrome	75	android	9	5
xiaomi mi a2 lite	424x895	chrome	72	android	9	3

APPENDIX I: Android and iOS usage January-July 2020 and Variation 2020-2021

Table 16 - Android Usage from January 2020 to July 2020

Android Version	% on total number of accounts	2021 usage (percentual points)
9	46.09%	-25.92
8	21.43%	-9.17
7	11.65%	-6.12
10	7.20%	+33.04
6	6.78%	-3.31
5	4.97%	-2.98
4	1.87%	N/A
Remaining Android Versions	0.01%	N/A
Total	**100%**	

Table 17 - iOS Usage from January 2020 to July 2020

iOS Version	% on total number of accounts	2021 usage (percentual points)
12	48.35%	-42.19
13	42.47%	+7.79
11	4.57%	-3.67
10	2.85%	-2.61
9	1.37%	N/A
Remaining iOS Versions	0.34%	N/A
Total	**100%**	

APPENDIX J: Devices per QA

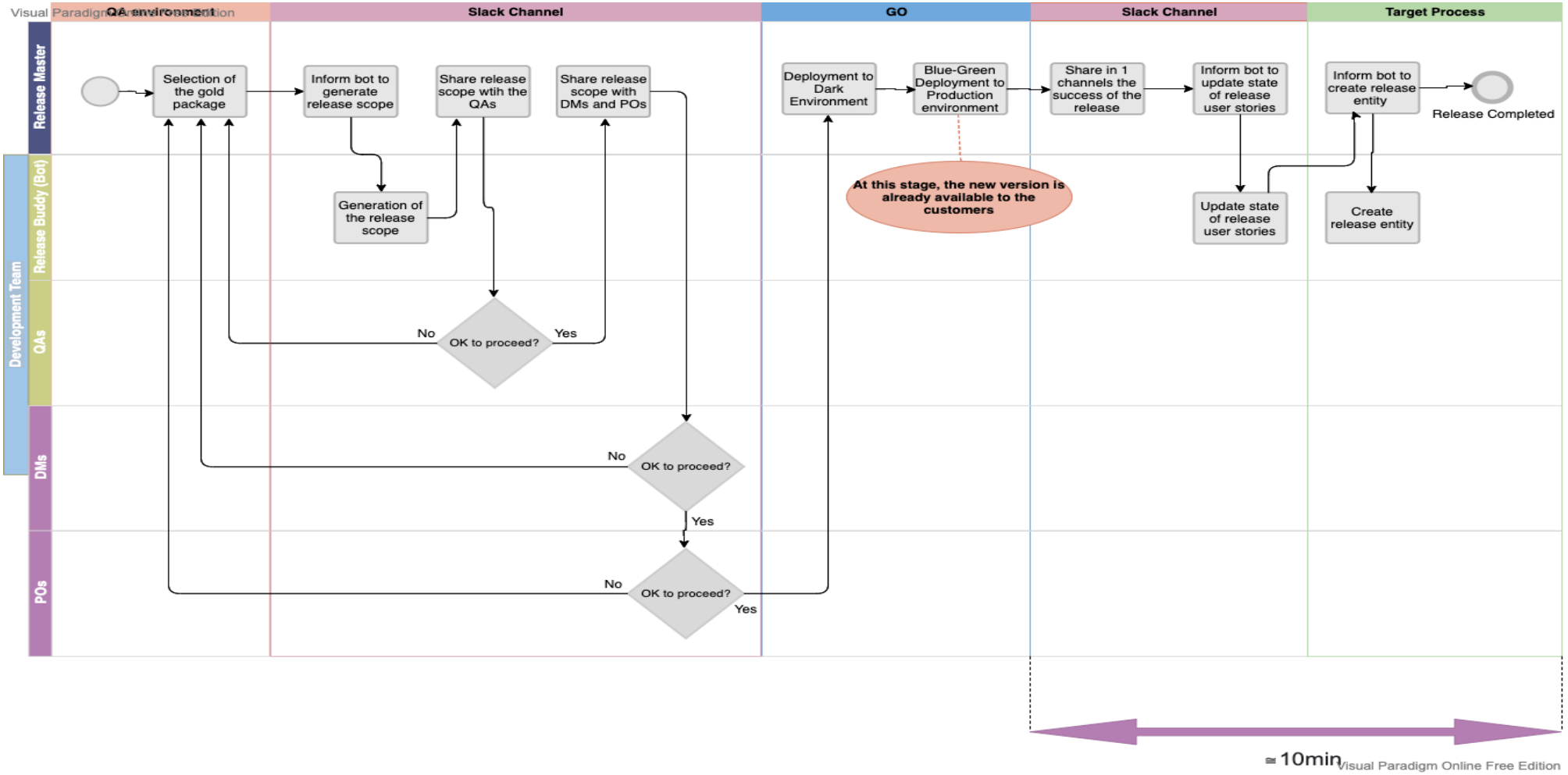
Table 18 - iOS Devices per QA

QA	Device Model	iOS Version
QA1	iPhone 12 Mini	14.6
QA2	iPhone 11	14.0.1
QA3	iPhone XS Max	13.6.1
QA4	iPhone 8	14.4
QA5	iPhone SE (2nd generation)	13.3.1

Table 19 - Android Devices per QA

QA	Device Model	Android OS Version
QA6	Samsung Galaxy S21	11
QA7	Samsung Galaxy S20	10
QA8	Samsung Galaxy S9	9
QA9	Samsung Galaxy A20e	10
QA10	Huawei P30 Pro	11
QA11	Xiaomi Redmi 8A	10
QA12	10Motorola Moto G9 Plus	

APPENDIX K: Release Process Map after changes



APPENDIX L: Severity Classification and Priority

Priority	Summary	Description
P1	A Product or Business Service is unavailable to Paddypower Betfair customers.	Any incident which causes a Product or critical functionality to become unavailable: Customers are unable to transact (login, fund, bet, register) with Paddypower or Betfair.
		Critical functionality can include, but is not limited to:
		Bet Placement, Login, Account funding
		Viewing of (all) markets
		Ability to manage markets.
		Key Game Products not loading
		Breach of our regulatory requirements or result in a Security risk (such as breaching ISO 27001)
P2	A Product or Business Service is severely degraded, but is still available to Betfair customers.	A Product or Business Service is performing in a severely degraded capacity, but is still available. Important functionality may be unavailable, but not preventing users from accessing critical functions. Important functionality can include, but is not limited to:
		Betting History or P&L
		Markets settlement
		Games or Sports Products intermittently returning 500 errors
		Content in highly visible areas
P3	A Product or Business Service has a significant issue but does not require the Major Incident response.	A Product or Business Service has a significant issue, or an issue with the potential to become a P1/P2 if unresolved. It does not require the Major Incident process to be invoked but requires immediate technical attention.
		Issue does not impact customers or colleagues access to core products or critical functionality
		Issue only impacts a relatively small number of users, or a single VIP user, or minor market(s)
		Settlement failure on some markets only
		Missing Silks in the run-up to Horse Racing
		Loss of resiliency on core systems
P4	A Product or Business Service with no/limited impact to customers, or impacts single users.	A Product or Business Service has a non-significant issue which does not require an immediate response but needs to be corrected.
P5	A Product or Business Service with no/limited impact to customers, or impacts single users.	An incident which may or may not need to be resolved at a future date - it may be assessed for relative value vs effort.

APPENDIX M: Release Process Checklist

Area	Severity Classification	Dark and Production Validations		<Device Model> <OS Version> <Responsible QA Name>
Login	P1	Login		☐
Betslip	P1	SBK - Insert a stake below of minimum value and check if the error message is displayed		☐
	P1	SBK - Insert a stake above of maximum value and check if error message is displayed		☐
	P1	SBK - Place a bet		☐
	P3	SBK Multiples - Add some selections to the betslip and check if the odds movement is working		☐
	P1	SBK Multiples - Insert a stake below of minimum value and check if the error message is displayed		☐
	P1	SBK Multiples - Insert a stake above of maximum value and check if error message is displayed		☐
	P1	SBK Multiples - Place a bet with a stake value higher than the account balance and check if error message is displayed		☐
	P2	SBK Multiples Free Bets - Place bet using bonus for multiples		☐
	P1	SBK Multiples - Place a multiple bet		☐
	P1	EXC - Place a matched bet		☐
	P1	EXC - Insert a stake value higher than the account balance and check if error message is displayed		☐
	P1	EXC - Place an unmatched bet		☐
	P1	EXC - Edit an unmatched bet and insert an invalid price increment; check if the error message is displayed and if the price value auto-corrects		☐
	P1	EXC - Edit an unmatched bet with success and check if the receipt values are updated		☐
	P1	EXC - Place a partially matched bet		☐

	P1	EXC - Edit a partially matched bet and insert an invalid price increment; check if the error message is displayed and if the price value auto-corrects		☐
	P2	EXC - Place bet using Free Bets bonus		☐
	P1	EXC - Cancel an unmatched bet		☐
	P1	EXC - Validate PnL: check if PnL and What-if appear when changing runner's size		☐
	P3	Trash a runner from betslip		☐
Cash-out	P1	EXC – Cash-out a Bet		☐
Search	P2	Search (teams, events, etc) on search page and navigate to the searched item		☐
	P4	Navigate to a Sports Page using the AZ Menu		☐
Market Rules	P2	Go to a market rules and analyse the information		☐
Market Graphs	P3	EXC - Click on runner and validate if market graphs are displayed		☐
Regulatory Footer	P1	Check if the footer is displayed on the bottom of the pages		☐
Logout	P2	Logout		☐