MESG

MESTRADO EM ENGENHARIA DE SERVIÇOS E GESTÃO

Análise e Melhoria de Processos de Gestão de Equipas de Software

Philippe Ribeiro Crétu

Dissertação de Mestrado

Supervisor na FEUP: Prof. Mário Amorim Lopes

Supervisor na E-goi: Ivo Pereira

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

2020-07-03

Resumo

Para a garantia de resultados satisfatórios na gestão de um projeto de desenvolvimento de software, é necessário a utilização de práticas e ferramentas que melhorem a eficácia da empresa, considerando o custo, a qualidade e a rapidez na entrega do serviço. Este tipo de projetos exige boas práticas de engenharia, que forneçam soluções sustentáveis, de qualidade e usabilidade.

A presente dissertação teve como fundamento a melhoria da gestão de equipas de desenvolvimento de software, desde o levantamento de necessidades até ao seu lançamento e foi dividida em quatro etapas, cada uma com os seus objetivos intermédios. A etapa inicial passou pela compreensão do funcionamento do departamento de desenvolvimento de produto da empresa, designadamente da sua estrutura, processos e metodologia de trabalho, através de entrevistas aos colaboradores da empresa, da documentação existente e observação direta dos participantes no processo. A informação recolhida foi fundamental para o levantamento de processos que, consequentemente, foi parte do fio condutor para a etapa seguinte, que consistiu na identificação dos principais problemas e suas respetivas causas.

Constatou-se que os processos com maior necessidade de serem melhorados foram os de definição de objetivos, planeamento e lançamento, devendo-se essencialmente à sua ausência de uniformização, que conduziu a um desalinhamento de práticas entre as equipas, à falta de ferramentas de suporte e de sistematização das existentes, e à inexistência de contacto direto com os clientes para o desenvolvimento de novas funcionalidades, prejudicando o desempenho da empresa. Assim, na etapa seguinte, recorrendo a boas práticas e ferramentas de Engenharia de Serviços, foram formuladas sugestões de melhoria que abordaram cada um destes aspetos, nomeadamente a normalização dos processos, desenvolvimento e melhoria de ferramentas sustentáveis para o enquadramento da gestão de equipas de software, e a integração do cliente através da metodologia *Design Thinking*. Por fim, foram discutidos os possíveis resultados e limitações, visto que são melhorias cujo seu impacto só pode ser medido a longo prazo.

Abstract

To ensure satisfactory results in the management of a software development project, it is necessary to use practices and tools that improve the effectiveness of the company, considering the cost, quality and speed of service delivery. This type of projects requires good engineering practices that provide sustainable, quality and usability solutions.

This dissertation was based on the improvement of software development teams management, from the needs gathering to its launch, and was divided in four stages, each one with its intermediate objectives. The initial stage involved understanding the operation of the company's product development department, namely its structure, processes and work methodology, through interviews with the company's employees, existing documentation and direct observation of the participants in the process. The information gathered was fundamental for the survey of processes, which, consequently, was part of the guiding thread for the next stage, which consisted in identifying the main problems and their respective causes.

It was verified that the processes most in need of improvement were those of objective definition, planning and deployment, due essentially to their lack of standardisation, which led to a misalignment of practices among the teams, the absence of support tools and systematisation of the existing ones, and the lack of direct contact with the customers for the development of new features, harming the performance of the company. Thus, in the next stage, suggestions for improvement were formulated that addressed each of these aspects, namely the standardisation of processes, the development and improvement of sustainable tools for the management of software teams, and the integration of the client through the Design Thinking methodology. Finally, possible results and limitations were discussed, since they are improvements whose impact can only be measured in the long term.

Agradecimentos

Em primeiro lugar, gostaria de agradecer à minha família, pela motivação e apoio incondicional que me deram ao longo da vida para cumprir os meus objetivos.

Agradeço a todos os colaboradores da E-Goi pelo rápido acolhimento e integração que me propocionaram.

Um especial agradecimento a todos os membros do departamento de desenvolvimento de produto, pelo facto de terem sido as pessoas da empresa com quem colaborei mais, e pelo tempo e disponibilidade para partilharem comigo maior parte dos conhecimentos que necessitei para compreender o funcionamento da empresa e os aspetos cruciais para o desenvolvimento da dissertação.

Agradeço ao meu orientador, professor Mário Amorim Lopes por todo o contributo dado ao longo do projeto, nomeadamente na partilha de conhecimento, nas sugestões e diretrizes dadas, que foram cruciais para o cumprimento dos objetivos.

Por último, mas não menos importante, gostaria de expressar a minha gratidão à minha namorada e aos meus amigos próximos, que ao longo dos anos estiveram sempre presentes para me apoiarem nos momentos mais difíceis e por me proporcionarem momentos inesquecíveis.

Índice de Conteúdos

1 Introdução 1
1.1 Descrição e Motivação do Problema 1
1.2 Apresentação da empresa – E-goi 1
1.3 Objetivos e Questões de Pesquisa 2
1.4 Metodologia 2
1.4.1 Possíveis abordagens 2
1.4.2 Metodologia Selecionada 3
1.5 Estrutura 4
2 Enquadramento Teórico 5
2.1 Metodologias de Desenvolvimento de Software 5
2.1.1 Metodologias Tradicionais 5
2.1.2 Metodologias Ágeis 6
2.2 Lean aplicado ao desenvolvimento de software 11
2.2.1 Ferramentas Lean para o desenvolvimento de Software 13
2.3 Envolvimento do Cliente no Desenvolvimento de Software 14
2.3.1 Design Thinking 15
2.4 Gestão de Processos de Negócio 16
2.4.1 Business Process Management Notation 16
3 Descrição e Caracterização do Caso de Estudo 18
3.1 Estrutura do Departamento de Desenvolvimento de Produto 18
3.1.1 Descrição das Equipas de Produto 19
3.1.2 Cargos Operacionais e Respetivas Funções 21
3.2 Análise dos Processos de Gestão de Equipas de Produto 22
3.2.1 Processo da (Re)Definição de Objetivos 23
3.2.2 Processo de Planeamento 24
3.2.3 Processo de Implementação 25
3.2.4 Processo de Lançamento 25
3.2.5 Monitorização das Equipas de Produto 26
3.3 Metodologia de trabalho da empresa 27
3.3.1 Criação de Issues 27
3.3.2 Workflow do quadro Kanban 27
4 Identificação e Caracterização do problema 29
4.1 Problemas no Processo de Definição dos Objetivos 29

4.2 Problemas no Processo de Planeamento 30
4.3 Problemas no Processo de Lançamento 33
4.4 Considerações Adicionais 34
5 Identificação e Desenho de Soluções 37
5.1 Solução para o processo de Definição dos Objetivos 37
5.2 Soluções para o processo de Planeamento 37
5.3 Solução para o processo de Lançamento 41
5.4 Solução para problemas transversais ao Departamento de Desenvolvimento 41
6 Conclusões e perspetivas de trabalho futuro 46
6.1 Limitações e perspetivas de trabalho futuro 47
Referências 49
ANEXO A: Guiões de Entrevistas 51
ANEXO B: Matriz RACI *As-Is* 53
ANEXO C: Processo de Definição dos Objetivos 54
ANEXO D: Fluxo do Processo de Planeamento 55
ANEXO E: Fluxo do Processo de Implementação 56
ANEXO F: Fluxo do Processo de Lançamento 57
ANEXO G: Fluxo do Processo de Monitorização das Equipas 59
ANEXO H: Melhoria do Issue Creator 60
ANEXO I: Fluxo com novo Estado 63
ANEXO J: Issue Resolutions 64

Lista de Figuras

Figura 1 - Utilização de metodologias de desenvolvimento ágeis (VersionOne, 2019) .. 7
Figura 2 - Framework Scrum (Rubin, 2012) 10
Figura 3 - Estrutura do departamento de desenvolvimento de produto da E-goi 18
Figura 4 - Mapa de Processos As-Is 22
Figura 5 - Quadro Kanban da E-goi 28
Figura 6 – Fluxo do Issue Creator As-Is 38
Figura 7 – Fluxo Issue Creator To-Be 38
Figura 8 - Sistema de identificação de duplicação de *issues* 40
Figura 9 - Mapa de Processos *To-Be* 42
Figura 10 - Matriz RACI *To-Be* 43

Lista de Tabelas

Tabela 1 – Cargos operacionais e respetivas funções..... 21
Tabela 2 - Atributos publicados no *Bitrix24* da nova versão 26
Tabela 3 - Resumo dos problemas identificados no processo de Definição dos Objetivos 30
Tabela 4 - Resumo dos problemas identificados no processo de Planeamento..... 32
Tabela 5 - Resumo dos problemas identificados no processo de Lançamento 34
Tabela 6 - Resumo dos problemas transversais..... 36
Tabela 7 - Técnicas para cada fase do processo de Design Thinking (Tschimmel, 2012) 44
Tabela 8 - Correspondência entre os problemas e as ações..... 45

Lista de Abreviações e de Siglas

DT – Design Thinking

JIT – Just in Time

KPI – Key Performance Indicator

OKR – Objective Key Result

PDCA – Plan, Do, Check, Act

PM – Product Manager

QA – Quality Assurance

RM – Release Manager

RACI – Responsible, Accountable, Consultant, Informed

UX – User Experience

1 Introdução

1.1 Descrição e Motivação do Problema

O projeto foi desenvolvido numa empresa cuja atividade principal envolve o desenvolvimento de uma plataforma de automação de marketing que possibilita aos clientes gerir as suas campanhas de marketing digital em diversos canais de comunicação.

Com o crescimento da empresa em termos de volume de negócios, sendo que todos os dias são criadas 350 novas contas na E-goi, torna-se crucial melhorar todo o processo de desenvolvimento. Tal como o aumento de volume de negócios, a empresa focou-se na melhoria do produto, proporcionando um maior número de funcionalidades e soluções customizadas, respondendo à crescente competitividade do mercado.

Estes fatores originaram o aumento gradual da complexidade do produto, pelo que surgiu a necessidade de investir em recursos humanos. No entanto, este crescimento originou problemas relativos à organização das diferentes equipas do departamento de desenvolvimento de produto, contribuindo para o aumento da complexidade interna, nomeadamente a falta de normalização dos processos transversais ao departamento, o desalinhamento de práticas entre as equipas e a falta de ferramentas sistematizadas que facilitem a gestão das atividades durante o ciclo de desenvolvimento.

Posto isto, o objetivo da presente dissertação está alinhado com os objetivos internos da empresa, atentando as tendências no mercado atual. Como tal, é necessário realizar o mapeamento e análise dos processos de departamento de desenvolvimento de produto desde a entrada de necessidades até ao lançamento de novas funcionalidades. Tendo estes processos em consideração, pretende-se avaliar futuras oportunidades de melhoria, de forma a aumentar a eficiência e eficácia no lançamento de novos incrementos no mercado.

1.2 Apresentação da empresa – E-goi

Este projeto foi desenvolvido na E-Goi, uma empresa portuguesa fundada em 2008, localizada em Matosinhos, que desenvolve soluções e produtos, fornecendo aos seus clientes uma melhor gestão de conteúdo de marketing digital. Esta plataforma é definida como um software multicanal de marketing de automação, incluindo *E-mail*, SMS, *SmartSMS*, *Voice notifications*, *Push notifications* e *WebPush*. Esta comunicação é designada para diferentes segmentos de clientes que pretendem usar a plataforma para captar e processar o comportamento dos clientes/subscritores, permitindo-os reagir automaticamente às suas ações e adaptar o conteúdo e formato promocional em função das preferências do cliente final. Adicionalmente, a base de dados do utilizador pode ser segmentada e gerida, mediante o comportamento e dados analíticos fornecidos, através de relatórios disponíveis na plataforma. Tendo em conta as características do negócio, atualmente esta empresa tem mais de 100 empregados e serve mais de 500.000 clientes espalhados por 40 países diferentes.

A visão da empresa é alcançar uma rede que relaciona os clientes finais com as marcas, criando sinergias que permitam uma maior satisfação e criação de valor. A sua missão é criar soluções de comunicação digital, acessíveis a todos os profissionais de marketing espalhados pelo

mundo, de forma a maximizar resultados, com base em práticas de respeito pelos clientes e pela sua privacidade.

O modelo de negócio da E-goi é baseado num SaaS, fornecendo um software de automação de marketing, em troca de uma comissão paga pelo modelo usado. Este ano o plano de negócio está a ser reformulado, devido aos problemas do modelo atual. O novo modelo será composto por quatro planos - *Social Share, Starter, Pro* e *Corporate*-, cada um com as suas condições e benefícios.

O *Social Share* é um plano grátis, mas tem bastantes limitações devido a não conter base de dados, no entanto permite a gestão de redes sociais através da criação e agendamento de publicações. O *Starter* vai conter todas as funcionalidades do *Social Share* com acréscimo de uma lista de contatos e número de SMS limitados. Relativamente ao plano *Pro*, os utilizadores poderão usufruir de todo o potencial da plataforma e com mais recursos em comparação com o modelo *Starter*. No que se refere ao plano *Corporate*, continuará igual, visto que foi criado especificamente para atender às necessidades dos clientes corporativos. Neste novo modelo, será possível a customização das funcionalidades disponíveis em cada plano, o que permite uma maior flexibilidade para com as necessidades do cliente, apesar de terem de pagar um valor extra por cada acréscimo de recursos.

A necessidade do desenvolvimento de projeto advém da existência de competidores fortes neste mercado, requerendo constante inovação, tomando atenção às carências dos clientes. A empresa está em constante crescimento, devido ao aumento diário de utilizadores na plataforma, surgindo desafios contínuo e oportunidades para o melhoramento da performance

1.3 Objetivos e Questões de Pesquisa

Após compreender as motivações e o foco do projeto, o próximo passo incluiu a seleção dos objetivos a serem executados, nomeadamente:

(i) O1: Análise e levantamento de todo o processo de desenvolvimento de software da empresa;
(ii) O2: Identificação dos processos críticos;
(iii) O3: Auditoria e mapeamento dos processos chave;
(iv) O4: Implementação do(s) processo(s) melhorado(s);

Também a definição das questões de pesquisa foi fundamental para a organização do trabalho ao longo do projeto. Neste sentido, esta dissertação pretende responder às seguintes questões de pesquisa:

(i) Questão de Pesquisa 1 (QP1): Quais são os gargalos atuais nos processos de gestão de equipas de software?
(ii) Questão de Pesquisa 2 (QP2): Quais as soluções que se adequam aos gargalos identificados?

1.4 Metodologia

1.4.1 Possíveis abordagens

A pesquisa qualitativa implica a seleção de como medir fatores relevantes e quais as técnicas de pesquisa que serão aplicadas, incluindo, como exemplo, inquéritos e testes. Uma vez selecionada, o investigador registará e verificará toda a informação recolhida. Através dos

dados quantitativos é possível ter uma correlação entre as variáveis e os resultados (Choy, 2014).

A pesquisa quantitativa é vantajosa no sentido de poder ser administrada e avaliada rapidamente. Por outras palavras, há pouco consumo de tempo durante a preparação da pesquisa e a obtenção de uma resposta pode ser relativamente rápida. Além disso, esta abordagem permite a comparação entre organizações e outros grupos, como a maior facilidade de alcançar a opinião dos diferentes entrevistados (Choy, 2014). No entanto, o ênfase nos dados e a incapacidade de obter informação enriquecedora são pontos fracos da pesquisa quantitativa, uma vez que ela não fornece perceções e descrição da experiência (Taylor, 2005).

Em contraste, a pesquisa qualitativa é uma abordagem indutiva com foco na descrição de múltiplas realidades, sendo possível desenvolver uma melhor percetibilidade e capturar as diferentes perspetivas do ser humano (Taylor, 2005). Com este tipo de exploração é possível uma melhor compreensão dos comportamentos, crenças e suposições num contexto em particular. Adicionalmente, é aplicada com base na "lógica prática", não existindo um caminho de pesquisa linear (Choy, 2014). A observação, casos de estudo, entrevistas e recolha de informação em documentação existente, são as diferentes formas de recolha de dados utilizados nesta pesquisa (Taylor, 2005).

Com base em Taylor (2005), os métodos qualitativos foram concebidos para envolver o investigador direta ou indiretamente no processo. Os dados obtidos não são representados por número, mas sim pela interpretação e envolvimento que o investigador teve fazer ao longo do processo (Taylor, 2005). Contudo, o resultado obtido não é objetivo e é necessário ter competências para entrevistar, para recolher a informação mais importante num curto período de tempo (Choy, 2014).

A grande diferença entre estas duas pesquisas é a natureza dos dados. Enquanto na pesquisa quantitativa, há uma extensa recolha de dados na forma de números, a pesquisa qualitativa é baseada em impressões, palavras, frases, fotos, entre outros, ditando diferentes estratégias de pesquisa (Choy, 2014).

De acordo com as metodologias escolhidas para os diferentes casos de estudo por Choy (2014), verificou-se que, independentemente da metodologias de pesquisa escolhida, haverá sempre impactos positivos e negativos sobre os resultados. Assim, os pontos fortes e fracos de ambas as metodologias continuam a ser compreendidos e antecipados. Portanto, o uso de ambas as metodologias para a pesquisa é vantajoso, devido a complementarem-se (Choy, 2014).

1.4.2 Metodologia Selecionada

Este projeto iniciou-se com a pesquisa literária sobre as áreas de conhecimento mais importantes. Concluída esta fase, foi filtrada a informação mais relevante, tendo sido mencionados os melhores recursos, que permitiram acrescentar conhecimento e valor à mesma. Portanto, por meio dos conhecimentos teóricos e do estado de arte abordados na revisão literária, perceberam-se os principais problemas em contextos particulares durante o desenvolvimento de software, com o objetivo de entender quais são os mais frequentes, permitindo uma análise dos processos com outra perspetiva.

Depois disto, optou-se por desenvolver um estudo do tipo qualitativo para perceber todo o processo de desenvolvimento de software, e neste caso em particular, a melhoria da gestão das equipas, pelo que foi necessário obter respostas e perceber a experiência daqueles que trabalham

na área de gestão de produtos de software. Assim, foi necessário interpretar a realidade e a experiência, de modo a ter uma direção correta de como abordar assertivamente o problema. Foi possível observar o dia-a-dia dos colaboradores e identificar abordagens que possam auxiliar na melhoria de performance da empresa. Além disso, para levantar todos os processos do departamento de desenvolvimento, foram criados três guiões para a realização de entrevistas semiestruturadas aos gestores das equipas core, gestores de equipa de produto e restantes departamentos da empresa, que se encontram descritos no *Anexo A.*

Após a análise dos processos da empresa, foi utilizada uma abordagem dedutiva, para a identificação de oportunidades de melhoria e as respetivas causas, através da aplicação de conceitos e teorias já desenvolvidas pelos autores referenciados. Nesta fase, as entrevistas realizadas aos diferentes membros da empresa auxiliaram à complementação dos problemas identificados.

Por fim, foram desenhadas soluções para colmatar os problemas anteriormente identificados, recorrendo ao uso de métodos de Engenharia de Serviços. Tanto o conhecimento derivado do contexto teórico, como a realização de workshops com os colaboradores do departamento de desenvolvimento da empresa, foram fundamentais para a implementação das soluções.

1.5 Estrutura

No presente capítulo é feita uma contextualização do tema, uma descrição do problema em questão e apresentadas as motivações, os objetivos esperados com este trabalho. Além disso, são referidos possíveis exemplos de métodos e técnicas para recolher e analisar dados e descrito qual a metodologia selecionada para o desenvolvimento deste projeto.

O capítulo 2 serve de clarificação e enquadramento teórico os temas inerentes a esta dissertação, sendo eles as Metodologias de Desenvolvimento de Software, Lean Aplicado ao Desenvolvimento de Software, Gestão de Processos de Negócio e Envolvimento do Cliente no Desenvolvimento de Software.

No terceiro capítulo é realizada uma análise do departamento de desenvolvimento da empresa utilizada como caso de estudo e está estruturado em três subsecções: a primeira consiste na descrição da estrutura do departamento de desenvolvimento de produto; a segunda parte compreende uma análise e caracterização do modelo *As-Is*; a terceira subsecção refere-se à metodologia de trabalho utilizada pela empresa.

O capítulo quatro é dedicado à identificação de oportunidades de melhorias e as causas associadas.

No quinto capítulo são apresentadas propostas de ações a aplicar de forma a colmatar os impactos negativos identificados no capítulo quatro.

Por último, no capítulo seis são discutidas as conclusões e perspetivas futuras e ainda o contributo final do projeto que mitigaram as falhas identificadas.

2 Enquadramento Teórico

O presente capítulo tem a finalidade de explorar sucintamente os aspetos fundamentais das metodologias de desenvolvimento de software, referindo os prós e contras e os modelos mais utilizados pelas empresas neste setor.

De seguida são apresentados os conceitos Lean na vertente do desenvolvimento de software, e técnicas que suportam a sua aplicação.

Posteriormente é realizada uma revisão sobre a importância da colaboração do cliente durante o desenvolvimento de software, paralelamente à metodologia de *Design Thinking*.

Por fim, é apresentada a prática de Gestão de Processos de Negócio, referindo os benefícios da sua utilização e a respetiva notação.

2.1 Metodologias de Desenvolvimento de Software

As metodologias de desenvolvimento de software estão em constante evolução, devido ao surgimento de novas tecnologias e à necessidade de entrega aos clientes de produtos com maior complexidade. Consequentemente, as organizações tecnológicas têm de se adaptar a este ambiente empresarial dinâmico para terem a capacidade de competirem nesta indústria (Nerur & Balijepally, 2007).

Segundo Ramesh, Cao, and Baskerville (2010), a engenharia tradicional de requisitos é muito diferente do ambiente empresarial em constante mudança, na qual operam cada vez mais organizações. Portanto, as metodologias de software ágeis têm revolucionado positivamente o processo de desenvolvimento de software e têm enfrentado com sucesso muitos desafios atuais, tais como, rápidas mudanças dos concorrentes em eventuais ameaças competitivas, preferências dos *stakeholders*, pressão de *time-to-market* e tecnologia de software (Ramesh et al., 2010).

2.1.1 Metodologias Tradicionais

Relativamente às metodologias de desenvolvimento tradicionais, Moniruzzaman and Hossain (2013) afirmam constituírem processos sequenciais e pesados, considerando o extenso planeamento e visão a longo prazo. A fase de planeamento nestas metodologias antecipa o prazo para uma determinada tarefa e o orçamento necessário através de um design detalhado como o sistema deve ser desenvolvido, incluindo a distribuição de tarefas entre os membros da equipa (Awad, 2005). Normalmente 30% do tempo gasto é requisitado para a fase de planeamento, que inclui a definição e design dos processos, originando uma grande quantidade de documentação. Esta abordagem não é efetiva o suficiente, devido a ser impossível alcançar uma especificação de requisitos clara, consistente e completa antes da implementação (Ramesh et al., 2010). Além disso, a mudança de requisitos durante o ciclo de desenvolvimento de software é inevitável, provocando grandes dificuldades, especialmente alterações posteriores à implementação dos requisitos (Nurmuliani, Zowghi, & Williams, 2006).

- ***Waterfall***

O Desenvolvimento *Waterfall*, conhecido em português como Desenvolvimento em Cascata, é um modelo tradicional que foi criado por Winston W. Royce para gerir desenvolvimento de produtos (Royce, 1987). Este modelo foi apropriado para projetos de desenvolvimento de

software, que consiste num processo sequencial, ou seja, é constituído por várias fases que têm de ser devidamente completadas pela ordem que foram planeadas. No entanto, o modelo *Waterfall* não inibe a possibilidade de retornar a uma fase anterior, contudo envolve custos maiores para recompor ou realizar alguma mudança de requisito (Munassar & Govardhan, 2010). As 6 fases fundamentais que compõem este modelo são a análise de requisitos, design, implementação, teste, lançamento e manutenção (Hibbs, Jewett, & Sullivan, 2009).

Nesta metodologia, os requisitos são claros e fáceis de implementar e são necessários poucos recursos para a execução. Para além disso, em cada fase existe documentação apropriada que tem de ser adotada para garantir a alta qualidade de software (Balaji & Murugaiyan, 2012). Por outro lado, o uso desta metodologia tem bastantes desvantagens devido a ser um processo linear, o que não permite que haja iterações entre as diferentes fases, ou seja, os problemas ao longo das fases nunca ficam totalmente resolvidos, resultando num sistema mal estruturado. Adicionalmente, este modelo é pouco flexível para alterar requisitos, não havendo a capacidade de responder a todas as necessidades dos *stakeholders* (Balaji & Murugaiyan, 2012).

2.1.2 Metodologias Ágeis

Em contraste às metodologias tradicionais, segundo Cho (2008), as metodologias Agile focam-se no desenvolvimento incremental e iterativo, através de um ciclo de vida leve e rápido. Similarmente, Aitken and Ilango (2013) referem que as metodologias ágeis regem-se por manter os programadores focados, remover os gastos e pela colaboração constante entre os programadores e os clientes. Para além disso, esta metodologia é compatível com a aproximação de um desenvolvimento com base em conceitos *Lean* (Aitken & Ilango, 2013).

Deste tipo de metodologias existe um vasto número de outras que se orientam pelos princípios que deram origem à metodologia ágil. São estas Scrum, Extreme Programming (XP), Desenvolvimento de Software Adaptativo (ASD) e Crystal. Contrariamente aos métodos tradicionais, o processo de desenvolvimento ágil, através das diferentes metodologias existentes, proporciona uma menor complexidade e risco e uma maior percetibilidade dos requisitos propostos pelos *stakeholders* devido ao seu constante feedback (Rahim, Chowdhury, & Das, 2017).

Na Figura 1, de acordo com o "13th Annual State of Agile Report", está representada a percentagem de uso das metodologias ágeis no mercado atual, revelando que 48% das empresas pesquisadas usa métodos ágeis em mais de metade das equipas, destacando-se o método Scrum devido à sua importância no mercado, sendo que 72% dos inquiridos usam os híbridos - "Scrum/XP Hybrid" e "ScrumBan" – ou o método original (VersionOne, 2019).

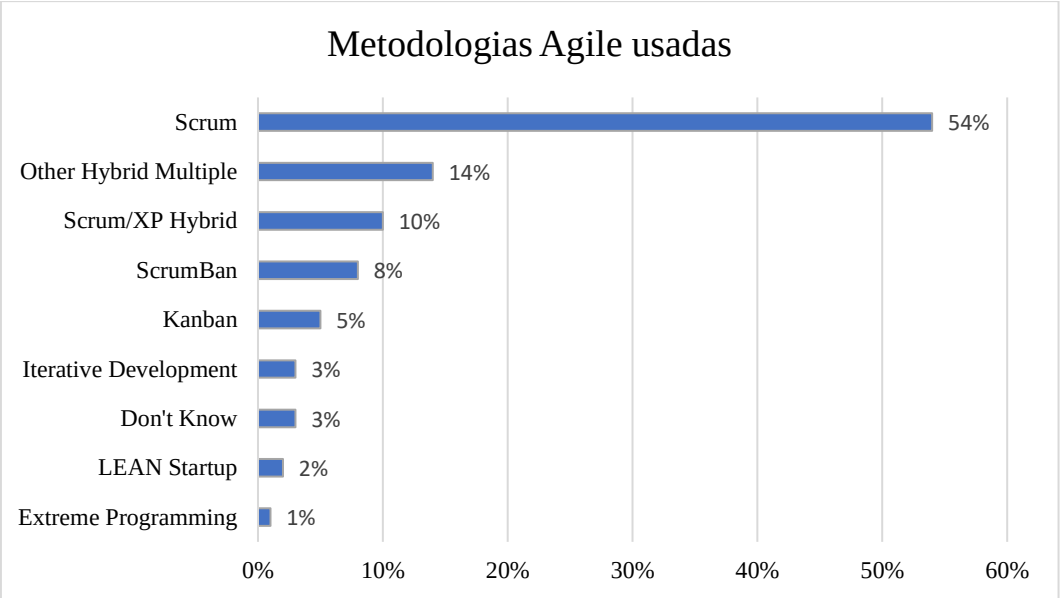


Figura 1 - Utilização de metodologias de desenvolvimento ágeis (VersionOne, 2019)

- **Scrum**

O Scrum, como referido anteriormente, é uma *framework* que se rege pelos princípios da metodologia ágil, tendo sido proposto por Ken Schwaber em 1995 e veio a responder a grandes necessidades na gestão de trabalho durante todo o desenvolvimento de software, incluindo a dinâmica de resposta às necessidades do cliente, complexidade dos requisitos de software, tempo limitado de entrega dos projetos, e às restrições de orçamento e de recursos(Scrum.org, 2018).

Esta *framework* foca-se na entrega de resultados de forma rápida de funcionalidades integradas, testadas e que tenham valor comercial. Com este foco, o Scrum auxilia as empresas a serem mais competitivas no mercado da tecnologia, no qual é necessário a rápida adaptação baseada no comportamento dos clientes, concorrentes e utilizadores, entre outros *stakeholders* (Scrum.org, 2018).

No entanto, o Scrum não resolve todos os problemas existentes na empresa, mas torna visível as perturbações e desperdícios que impedem a empresa de alcançar o seu verdadeiro potencial. Em termos de principais funções e responsabilidades destacam-se: *Product Owner*, Scrum Master, e a equipa de desenvolvimento (Cho, 2008).

O *Product Owner (PO)* tem a visão do produto e é responsável pela definição, priorização e comunicação dos requisitos do *Product Backlog* (Matharu, Mishra, Singh, & Upadhyay, 2015). Além disso, deve manter e comunicar claramente a visão do que é pretendido ser alcançado pela equipa. Portanto, é responsável pelo sucesso da solução que está a ser sustentada ou desenvolvida (Rubin, 2012).

A equipa de desenvolvimento é constituída entre 3 e 9 elementos, que são responsáveis por implementar as tarefas que são alocadas para cada *Sprint*, estipuladas pelo *Product Owner* (Matharu et al., 2015). As equipas devem ser autónomas e multidisciplinares para maximizar o desempenho da equipa (Cho, 2008).

O Scrum Master (SM) é responsável por assegurar que os valores, práticas e regras da metodologia ágil estão a ser decretadas e aplicadas devidamente (Cho, 2008). Complementarmente, este também presta auxílio na eliminação de impedimentos, tal como no apoio à equipa de desenvolvimento durante o próprio desenvolvimento das tarefas propostas (Matharu et al., 2015).

Na metodologia *Scrum,* há ainda que explicitar um conjunto de conceitos que lhe estão associados, e que a tornam diferenciada das restantes metodologias. A framework Scrum é constituída por 5 eventos: *Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective.* Estes eventos são usados para criar regularidade e minimizar a necessidade de reuniões que não estavam definidas. Para não haver desperdícios durante o processo, os eventos terminam somente quando o seu propósito for alcançado (Scrum.org, 2018).

De seguida encontra-se a explicação destes eventos em detalhe:

Sprint

Um *Sprint* é descrito como um período máximo de 1 mês, durante o qual é criado um incremento do produto – soma de todos os itens do *backlog* do produto concluídos durante uma *Sprint* e a soma dos incrementos de todas as sprints anteriores - "Concluído". Após a finalização de cada sprint, o incremento do produto já deverá estar disponível para utilizar. Cada *sprint* compreende um ciclo completo de produção, permitindo a previsibilidade e garantindo a inspeção e a adaptação do progresso em direção a um objetivo do sprint. Durante cada ciclo presume-se (Scrum.org, 2018):

(i) a não alteração que possa impactar negativamente o objetivo da *sprint;*
(ii) que os objetivos de qualidade não devem ser diminuídos;
(iii) o esclarecer o ajuste do objetivo entre o *Product Owner* e a equipa de desenvolvimento.

Sprint Planning

Um *Sprint Planning* tem uma duração máxima de oito horas, no qual é realizado um plano para que cada *Sprint* ocorra dentro do expectável. Durante este evento, é definido um objetivo chamado de *sprint goal*, no qual é suposto a equipa cumprir até ao final do *sprint*. Assim que o objetivo esteja definido, a equipa de desenvolvimento revê o backlog e categoriza com maior prioridade os itens que permitem cumprir o objetivo, estimando o esforço necessário para cada item, de modo à equipa ser capaz de os cumprir (Rubin, 2012). Para que o planeamento seja realizado com sucesso, este deve ser capaz de responder às seguintes duas questões (Scrum.org, 2018):

(i) "O que pode ser entregue no Incremento efetuado do próximo *Sprint*?"
(ii) "Como é que o trabalho escolhido para alcançar o incremento vai ser realizado?"

Daily Scrum

As *daily scrum* são de curta duração e têm como objetivo aprimorar a comunicação entre os membros da equipa de desenvolvimento, de modo a avaliar o estado de progresso da *Sprint* (Matharu et al., 2015). Conforme o estado do progresso, as atividades são sincronizadas através da criação de um plano para as próximas 24 horas. Com a realização desta reunião são identificados impedimentos existentes durante o desenvolvimento que são removidos, é

destacada e promovida a rápida tomada de decisões, e consequentemente a probabilidade de o objetivo da *sprint* ser alcançado e otimizado (Scrum.org, 2018).

Sprint Review

A *Sprint Review* ocorre no final de cada iteração *Sprint*, sendo o intuito a revisão do incremento e adaptação do *backlog* do produto, se necessário. Esta revisão é à base da colaboração entre a equipa de desenvolvimento e os *Stakeholders*, para perceber o que foi feito. Esta é uma reunião informal e visa incluir os seguintes elementos (Scrum.org, 2018):

(i) O *Product Owner* que faz uma introspeção das tarefas planeadas, incluindo as que foram concluídas e as que não foram;
(ii) A equipa de desenvolvimento evidencia e discute o que foi positivo e/ou negativo durante o *Sprint;*
(iii) A equipa de desenvolvimento apresenta o trabalho que foi concluído e o que não foi possível concluir;
(iv) O *Product Owner* discute o estado do *Product Backlog;*
(v) Todos os presentes definem o que é pretendido ser realizado no próximo *Sprint;*
(vi) Revisão de como as mudanças do mercado ou da potencial utilização do produto podem ter mudado e, naturalmente, o alinhamento das prioridades;
(vii) Revisão do horizonte temporal, orçamento, capacidade da equipa e mercado para os lançamentos do produto futuro.

Sprint Retrospective

O *Sprint Retrospective* ocorre após a *Sprint Review* e antes do *Sprint Planning*, com o objetivo de a Equipa *Scrum* analisar e apresentar um plano de melhorias para serem implementadas no próximo *Sprint.* Resumidamente, durante este evento a equipa discute o que correu bem na *Sprint* anterior, o que poderia ser melhorado e criar um plano para melhorar o desempenho do próximo *Sprint*, considerando o que falhou anteriormente (Scrum.org, 2018).

De maneira a representar o trabalho ou valor para fornecer transparência e oportunidades de inspeção, o Scrum é também constituído por 3 artefactos: *Product Backlog*, *Sprint Backlog*, *Increment* (Scrum.org, 2018).

Product Backlog

O *Product Backlog* é uma listagem ordenada de todas as tarefas que foram analisadas para implementar no produto. Todas as mudanças a serem realizadas no produto encontram-se nesta lista. Como referido anteriormente, é o *Product Owner* que é responsável por este artefacto, incluindo o seu conteúdo, disponibilidade e ordenação. A lista do *Product Backlog* é dinâmica, por haver a necessidade de responder ao ambiente em que o produto está envolvido, para este seja apropriado, útil e competitivo (Scrum.org, 2018).

Sprint Backlog

O *Sprint Backlog* é composto por um conjunto de itens selecionados da lista do *Product Backlog* para o *Sprint*, com base nas prioridades definidas pelo *Product Owner*. Durante o *Sprint* a equipa de desenvolvimento é responsável por modificar o estado da tarefa no *Sprint Backlog*. Este artefacto oferece uma visão em tempo real do trabalho que está a ser produzido pela equipa de desenvolvimento (Scrum.org, 2018).

Incremento

O Incremento é a soma do conjunto de itens concluídos com sucesso do *Product Backlog* durante uma *Sprint* e do valor acrescentado às *Sprints* anteriores. Este artefacto, quando "Concluído", é um passo direcionado ao objetivo pretendido. O objetivo principal do Scrum é fornecer um incremento "Concluído" (Scrum.org, 2018).

Funcionamento do ciclo Scrum

Na Figura 2 está ilustrado todo o funcionamento do Scrum através do relacionamento entre os eventos e os artefactos existentes.

Figura 2 - Framework Scrum (Rubin, 2012)

Considerando que este ciclo se inicia com a visão do *PO*, que consiste no que se pretende desenvolver, é necessário que seja repartido em várias funcionalidades que posteriormente serão colocadas no *product backlog* para serem devidamente priorizadas, sendo que o número de itens existentes no *product backlog* é superior às que a equipa de desenvolvimento é capaz de desenvolver num só *sprint* (Rubin, 2012).

O *sprint* está representado na imagem pela seta que envolve todo o ciclo. Este tem início no *sprint planning*, engloba o desenvolvimento durante o *sprint* (*sprint execution*) e acaba com a revisão e retrospetiva do mesmo (Rubin, 2012).

Para que o *sprint* seja cumprido com sucesso é importante que a definição do objetivo realizado durante *sprint planning* seja realista, considerando o tempo disponível, para não desmotivar a equipa de desenvolvimento. Para organizar o objetivo que está planeado, é criado um *sprint backlog*, que contém o plano com o detalhe de cada tarefa, como a equipa planeia desenhar, construir, integrar e testar o conjunto de funcionalidades (Rubin, 2012).

Durante o *sprint execution*, a equipa de desenvolvimento cria as funcionalidades selecionadas para o respetivo *sprint*. No início de cada dia é realizado um evento chamado *daily scrum*, no qual todos os membros de equipa estão presentes para realizar um plano de forma a sincronizar, inspecionar e adaptar o trabalho que tem de ser desenvolvido. Quando o *sprint execution* finalizar, significa que a equipa de desenvolvimento produziu um potencial incremento, que representa uma parte da visão do *Product Owner* (Rubin, 2012).

Seguidamente, para concluir o *sprint* são realizados dois eventos, para inspecionar e adaptar. Em primeiro lugar é feita uma revisão, *sprint review*, tendo como objetivo a colaboração entre

a equipa Scrum e os *stakeholders* para inspecionar o que foi feito. No segundo evento, chamado *sprint retrospective*, a equipa Scrum inspeciona o processo usado para desenvolver o incremento (Rubin, 2012).

2.2 Lean aplicado ao desenvolvimento de software

O termo *Lean*, também conhecido como Sistema Toyota de Produção surgiu, no século XX, no Japão, pela *Toyota*, após a segunda guerra mundial (Ohno, 1988). Esta empresa, com o intuito de se tornar competitiva, pretendia produzir carros em grandes quantidades com baixo preço e responder às necessidades dos clientes com eficiência. Tendo em conta as limitações, devido à crise do país, a fábrica japonesa resolveu inovar e criar um método próprio para resolver os problemas de produção. Com o sucesso das práticas implementadas, estas foram adotadas em diversas áreas da indústria para eliminar desperdícios e garantir a qualidade, originando o conceito *Lean Thinking* (Womack & Jones, 1996).

O *Lean Thinking* ajuda a entender os princípios do Lean (Womack & Jones, 1996):

(i) Identificação do valor: é criado pelo produto, sendo, no entanto, percecionado pelo cliente. Tendo isso em conta, é fundamental ter uma compreensão clara das necessidades do cliente;
(ii) Cadeia de valor: é o conjunto de procedimentos necessários para levar a ideia de um produto até ao cliente final. É importante que o processo de produção flua continuamente;
(iii) Otimização do fluxo: consiste no mapa que identifica cada passo do processo e categorias de cada passo em termos do valor adicionado.
(iv) Sistema pull: As encomendas do cliente são geridas garantindo que nada é construído antes de ser necessário;
(v) Perfeição: Explorar a perfeição no processo, identificando e removendo constantemente os desperdícios.

No entanto, a metodologia Lean, aplicada ao desenvolvimento de software, é constituída por 7 princípios, sendo traduzidos em práticas ágeis e usados conjuntamente com outros métodos ágeis (Poppendieck & Cusumano, 2012).

- **Princípio 1: Eliminar os desperdícios**

O principal foco e princípio orientador do Lean consiste em identificar e remover os desperdícios do processo, sem afetar o valor fornecido ao cliente (Wang, Conboy, & Cawley, 2012). Neste contexto, o desperdício significa tudo aquilo que não entrega ou adiciona conhecimento ao produto final (Poppendieck & Cusumano, 2012).

As maiores causas de desperdício no processo de desenvolvimento de software são (Petersen, 2010; Poppendieck & Cusumano, 2012):

1º Trabalho parcialmente realizado: Trabalho em desenvolvimento que não tem valor até ser concluído;

2º Funcionalidades extra: Funcionalidade que foi desenvolvida, mas que não cria valor para o cliente, causando gasto de recursos desnecessários;

3º Movimentos: Os colaboradores têm que fazer movimentos desnecessários para comunicar e esclarecer as dúvidas;

4º Troca de tarefas: Perda de concentração e dificuldade em retomar as tarefas;

5º Atrasos: Risco de alguma componente do produto ficar obsoleto;

6º Defeitos: Resolução de *bugs* no produto de software;

7º Processos desnecessários: Etapas no processo que podem ser removidas.

Adicionalmente, o *Lean Thinking* auxilia na identificação e remoção dos respetivos desperdícios, classificando o trabalho em três categorias distintas: atividades que agregam valor, atividades que não agregam valor e atividades necessárias sem valor acrescentado (Wang et al., 2012). Por isso, é importante as empresas analisarem a cadeia de valor para perceberem a origem e a causa do desperdício existente (Poppendieck & Cusumano, 2012).

- **Princípio 2: Integrar qualidade**

A IBM de modo a melhorar a prática para obter uma boa qualidade de software, criou uma abordagem denominada como "*top-down programming*", consistindo na integração dos módulos de software no sistema geral, à medida que vão sendo realizados, em vez de ser no fim do desenvolvimento (Poppendieck & Cusumano, 2012). De modo a garantir a qualidade de software é essencial que não existam bugs. Segundo Middleton (2001), existem 5 princípios que melhoram este aspeto:

(i) melhoria de qualidade contínua;
(ii) trabalhadores com competência a assumir responsabilidades;
(iii) prevenção de defeitos, evitando deteção aleatória;
(iv) medidas de qualidade simples e visíveis;
(v) dispositivos automáticos de medição de qualidade.

- **Princípio 3: Criar conhecimento**

O terceiro princípio visa garantir que o conhecimento sobre o software seja criado ao longo do desenvolvimento, em vez de seguir um plano rígido de requisitos pré-definidos, estabelecidos no início do desenvolvimento. Este princípio surgiu devido à necessidade de resposta do feedback originado após as *releases*, que permite às equipas interpretar de uma forma rápida nova informação e realizar as mudanças apropriadas. Por isso, é importante evitar o problema e explorar a qualidade durante a produção, em vez de procurar e corrigir os erros futuramente (MacCormack, 2001). Segundo MacCormack (2001), existem 4 práticas que garantem uma boa aceitação do mercado:

(i) *Releases* breves com um conjunto mínimo de funcionalidades para os clientes avaliarem e darem feedback;
(ii) Integração diária de novo código de software e feedback sobre alterações de design;
(iii) Uma equipa e/ou líder com uma experiência ampla e instinto para tomar boas decisões;
(iv) Uma arquitetura modular que dê suporte à habilidade de adicionar novas funcionalidades.

Além disso, para facilitar a disseminação do conhecimento na empresa, este princípio defende as seguintes práticas (Hibbs et al., 2009; Petersen, 2010; Poppendieck & Cusumano, 2012):

(i) Contacto constante com o cliente, de forma a recolher o seu feedback para que o departamento entenda os objetivos a serem cumpridos;
(ii) Colaboração entre as diversas equipas envolvidas no produto de software, de modo a recolher feedback e prever formas de melhoria contínua;

(iii) Colocação de um diagrama, numa área comum, de modo a permitir visibilidade a todos os membros da empresa;

- **Princípio 4: Adiar comprometimentos**

A tomada de decisão deve ser realizada após serem reunidas as condições essenciais. No caso de no momento não ser necessário, ou possível, tomar uma decisão, é preferível o adiamento para recolher informações e conhecimento, para que a solução seja sustentada por factos e não unicamente por especulações (Poppendieck & Cusumano, 2012). No entanto, não é aconselhado esperar demasiado tempo, para não atrasar outras componentes do projeto. Adicionalmente, antes da tomada de decisão é importante realizar um estudo sobre as necessidades e as alternativas. Se possível, o ideal é o uso da prática *set-based design*, ou seja, testar as duas melhores alternativas e escolher a que mais satisfaz as necessidades do problema em questão (Hibbs et al., 2009).

- **Princípio 5: Entrega Rápida**

Este princípio pressupõe que a entrega do produto ao cliente seja no tempo adequado. Na área do desenvolvimento de software, as *releases* de produção sucedem-se frequentemente, tendo melhorado significativamente a sua qualidade. O desenvolvimento de software tornou-se mais ágil com a existência de um fluxo, no qual são concebidas entregas frequentes de incrementos com pequenas mudanças (Poppendieck & Cusumano, 2012).

Este processo iterativo é representado pelo modelo "*The Evolutionary-Delivery Model*". Este modelo consiste em repartir o projeto em diversas partes, desenhadas para entregar subconjuntos de funcionalidades do produto total. Com esta abordagem, a equipa tem uma maior flexibilidade na definição da direção, através da alteração do foco dos futuros projetos (Hibbs et al., 2009).

- **Princípio 6: Respeitar as pessoas**

Este princípio visa que haja confiança entre as diferentes equipas e encoraja o trabalho em grupo, ou seja, que todos os membros participem ativamente na tomada de decisões. Portanto é fundamental para qualquer implementação Lean persuadir os colaboradores na tomada de decisões, incentivando a auto-organização e preservando a confiança na equipa (Poppendieck & Cusumano, 2012).

- **Princípio 7: Otimizar o todo**

Por último, a otimização do todo engloba perceber a cadeia de valor, desde o momento em que o pedido é recebido, até ao momento em que ele é resolvido ou entregue ao cliente. Este conhecimento da cadeia de valor vai auxiliar na perceção do impacto que a otimização de um processo pode ter noutro. Por outras palavras, não adianta melhorar um processo se, porventura, existe outro processo a ser impactado negativamente devido a não ter capacidade de resposta à melhoria implementada(Petersen, 2010).

2.2.1 Ferramentas Lean para o desenvolvimento de Software

No seguimento da metodologia Lean, existem várias técnicas que dão resposta a alguns desafios desde o levantamento de necessidades até ao lançamento de uma nova versão de software. De seguida apresentam-se exemplos que aplicam os princípios descritos anteriormente, aplicados ao desenvolvimento de software.

Kanban

Kanban é uma metodologia que pode ser aplicada durante o desenvolvimento de software, que destaca a entrega JIT. Esta é constituída por toda a informação necessária relativa à produção/montagem de um produto. Em cada fase é detalhado o caminho para a sua conclusão, sendo que o planeamento é realizado com base na priorização de tarefas, definição do *workflow* e no tempo de entrega (Lei, Ganjeizadeh, Jayachandran, & Ozcan, 2017). Esta ferramenta quando aplicado no contexto de software é caracterizada pela entrega contínua de pequenas iterações, sendo permitidas mudanças a qualquer momento (Ahmad, Markkula, & Oivo, 2013) . De acordo com Lei et al. (2017), esta metodologia possui os seguintes 6 princípios aplicados ao desenvolvimento de software:

(i) Limitar o trabalho em progresso (WIP);
(ii) Tornar o processo de desenvolvimento visível;
(iii) Aumentar a produção;
(iv) Utilizar um *backlog* fixo;
(v) Integrar qualidade;
(vi) Obter valor ao longo do processo de desenvolvimento.

O método Kanban propõe a utilização de cartões ou post-its num quadro, que permitem tornar o processo de desenvolvimento visível, devido a cada elemento da equipa estar alocado a uma tarefa e ser possível comunicar as prioridades e possíveis gargalos. Adicionalmente, o objetivo é limitar o WIP, e desenvolver somente os itens que foram planeados fazer, de modo a permitir aos programadores focarem-se em poucos itens de cada vez. Ao limitar o WIP, é alcançado um ritmo sustentável de desenvolvimento, originando uma maior qualidade dos produtos e um aumento de performance da equipa. Através destes benefícios no fluxo é possível reduzir o tempo de execução, facilitando lançamentos regulares que ajudam a fortalecer a confiança com o cliente (Ahmad et al., 2013).

Scrum e Kanban são ambas abordagens de desenvolvimento ágeis que tornam o trabalho mais efetivo e descrevem o que é necessário realizar. No entanto, não são perfeitas nem completas, e apenas fornecem algumas diretrizes e limitações (Rubin, 2012).

Ciclo PDCA

O ciclo PDCA é uma metodologia de melhoria contínua, sendo que cada uma das letras representa uma fase a implementar. Surgiu para que na conclusão de cada uma das quatro etapas se inicie novamente um ciclo novo.

Este ciclo é composto pelas seguintes 4 fases (Sokovic, Pavletic, & Pipan, 2010):

(i) Planear (Plan) – definição dos objetivos e planeamento de dados e informação necessárias para as restantes etapas;
(ii) Fazer (Do) – implementar o plano definido na fase anterior;
(iii) Verificar (Check) – verificar os resultados da implementação, através de monitoramento de ações;
(iv) Agir (Act) – desenvolver ações corretivas, no caso de existirem falhas ou de não alcance dos resultados pretendidos.

2.3 Envolvimento do Cliente no Desenvolvimento de Software

O desenvolvimento de novos serviços visa refletir duas dimensões diferentes, nomeadamente a eficácia operacional e a competitividade no mercado. A eficácia operacional (velocidade de

inovação e qualidade técnica) traduz-se na implementação do trabalho, pelo qual é avaliado o esforço numa perspetiva interna. Enquanto a competitividade do mercado (superioridade competitiva e performance das vendas) é refletida pelo sucesso obtido, que é medido numa perspetiva externa (Carbonell, Rodríguez-Escudero, & Pujari, 2009).

Como referido no capítulo 2.1, de forma a responder às necessidades do mercado, as metodologias ágeis zelam pela colaboração por parte do cliente, o que se tornou um fator crucial durante o processo de desenvolvimento de software, nomeadamente na definição das *user stories*, discussão de novas funcionalidades para o produto, priorização de listas de funcionalidades e no rápido feedback fornecido à equipa de desenvolvimento (Hoda, Noble, & Marshall, 2011). Por isso, a recolha de informação baseado no feedback e experiências dos utilizadores é considerado um recurso essencial, pelo qual as empresas dependem para desenvolver produtos com maior êxito (Carbonell et al., 2009).

Em contraste, a falta de envolvimento do cliente nas metodologias ágeis pode originar consequências para as equipas multidisciplinares, inclusive problemas na recolha e esclarecimento de requisitos, problemas na insegurança de feedback, perda de produtividade e maior pressão por excesso de comprometimento (Hoda et al., 2011). Lindberg, Meinel, and Wagner (2011) afirmam que, por vezes, independentemente de as funcionalidades estarem tecnicamente bem desenvolvidas, podem-se revelar incompreensíveis na perspetiva do utilizador.

Portanto, o envolvimento do cliente é fundamental para uma melhor performance no desenvolvimento de serviços (Carbonell et al., 2009). Neste sentido emergiram metodologias que abordam padrões cognitivos para captar múltiplos conhecimentos e perspetivas dos clientes, com o propósito de transformar de forma criativa o know-how adquirido em novos conceitos de serviço e produto (Lindberg et al., 2011).

2.3.1 Design Thinking

De acordo com Tschimmel (2012), o *Design Thinking*, atualmente, não é somente um procedimento cognitivo, como também se tornou um conjunto de ferramentas para inovação de processos, que conecta a abordagem do design criativo ao pensamento empresarial tradicional, com o objetivo de planear e resolver problemas. Para compreender o funcionamento desta estratégia é necessário entender dois conjuntos de conceitos: o problema e o espectro da solução e, por outro lado, o pensamento divergente/convergente. O design aborda tanto o problema como a solução, algo a ser estudado através de um pensamento divergente e convergente (Lindberg et al., 2011).

Adicionalmente, o DT é centrado no ser humano para o desenvolvimento de novos serviços, que integra conhecimentos da área de design, ciências sociais, engenharia e negócios (Gurusamy, Srinivasaraghavan, & Adikari, 2016). Esta metodologia é dividida nas seguintes cinco etapas (Gurusamy et al., 2016):

(i) Empatia: obter uma visão geral do cliente tendo em consideração o contexto, incluindo as necessidades físicas, emocionais, a intenção, as motivações e os fatores ambientais que rodeiam;
(ii) Definição: considerando os dados recolhidos na fase anterior, definir o desafio que será enfrentado, clarificando todos estes dados;

(iii) Ideação: esta etapa é dedicada a criar ideias que poderão ser possíveis soluções, ou parte da solução, que resolvem os problemas identificados na fase da definição;
(iv) Protótipo: criar um protótipo que seja realista e permita a interação do utilizador, com o objetivo de responder a perguntas que o aproximam da solução final;
(v) Teste: esta etapa é destinada a recolher feedback dos utilizadores sobre os protótipos criados e conforme o retorno, redefinir, ou não, os protótipos para obter a solução final.

A aplicação desta metodologia no âmbito do desenvolvimento TI implica um tipo de pensamento distinto, que amplie as capacidades de resolução de problemas das equipas de desenvolvimento com o objetivo de tornar os seus resultados mais inovadores. Comparativamente com as metodologias ágeis, o DT tem algumas características em comum, tais como o foco no utilizador, desenvolvimento iterativo e colaboração entre os membros da equipa. No entanto, existem algumas divergências entre estas duas metodologias, sendo que as metodologias ágeis se focam no desenvolvimento incremental contínuo, evitando investigar e comparar caminhos para novas soluções (Lindberg et al., 2011). Nesse sentido, é evidente que a metodologia ágil contorna o pensamento divergente, limitando-se a seguir o plano estipulado. Além disso, apesar de haver colaboração entre as equipas, as pessoas envolvidas costumam ter um pensamento mais técnico e menos criativa do que no DT (Lindberg et al., 2011).

A título de exemplo da boa utilização desta metodologia neste âmbito, é o da *Salesforce Ignite*, pelo facto de usufruirem de técnicas para responder às necessidades dos seus clientes, de modo a melhorar as suas experiências e suportarem estrategicamente os seus negócios para retirarem o maior valor dos seus produtos (Beckman & Whalen, 2018).

2.4 Gestão de Processos de Negócio

Business Process Management (BPM), conhecido em português como Gestão de Processos de Negócio, é um conceito utilizado nas organizações para desenhar, executar, monitorizar e otimizar processos de negócio. Esta prática visa melhorar o processo de gestão da cadeia de valor através da gestão de eventos, atividades e decisões que adicionam valor à organização e aos seus clientes. Além desta disciplina, existem várias que ajudam a melhorar o desempenho operacional das organizações e que estão incorporadas no BPM, tais como *Total Quality Management (TQM), Lean, Six Sigma* e Gestão de Operações. O BPM tem as contribuições destas várias disciplinas relacionadas com a Tecnologia de Informação moderna, para garantir o alinhamento dos processos de negócio com as metas estratégicas e objetivos de desempenho das organizações (Dumas, La Rosa, Mendling, & Reijers, 2013).

2.4.1 Business Process Model Notation (BPMN)

BPMN define um diagrama que é baseado numa técnica de desenvolvimento de fluxos, através de esquemas gráficos, que representam operações de processos de negócio. O principal objetivo do BPMN foi criar uma linguagem comum, de maneira a representar os processos de negócio e ser legível para todos os gestores. Assim, ao desenhar um diagrama de processos, é possível usar símbolos universais que serão entendidos por diversos profissionais (White, 2004). Adicionalmente, de acordo com Ko (2009), esta modelação oferece diversos benefícios para as empresas, tais como a maior visibilidade e conhecimento das atividades da empresa, maior capacidade de identificação de gargalos e potenciais domínios de otimização, redução dos tempos de execução e melhor definição de funções e papéis na empresa.

Portanto, estes processos podem ser utilizados tanto para identificar e documentar os procedimentos atuais da empresa (modelo *as-is*) como para propor uma melhoria (modelo *to-be*). Estes são, normalmente, complementados com texto descritivo (Dumas et al., 2013).

Notação BPMN

O BPMN consiste na representação de atividades e no controlo de fluxo para definir a sua ordem de desempenho. A notação apresenta, para estes objetivos, um conjunto de elementos gráficos que a tornam mais intuitiva para todas as pessoas envolvidas no projeto. Estes elementos podem ser divididos nas seguintes 4 categorias (Chinosi & Trombetta, 2012):

(i) Objetos de fluxo;
(ii) Objetos de conexão;
(iii) *Swimlanes;*
(iv) Artefactos.

Objetos de fluxo

Relativamente aos objetos de fluxo, estes representam todas as ações que podem ocorrer dentro de um processo de negócio (Chinosi & Trombetta, 2012). São constituídos por eventos, atividades e decisões. Os eventos são representados por círculos, e consistem em acontecimentos que podem ter impacto num processo de negócio. As atividades são uma forma representativa de ações, estas são compostas por dois tipos, os subprocessos e as tarefas. Os subprocessos são funcionalidades que podem ser detalhadas, enquanto as tarefas não podem ser subdivididas em partes mais pequenas. As decisões são representadas por losangos, e servem para controlar o fluxo do processo de negócio, determinando as decisões pretendidas (White, 2004).

Objetos de conexão

O objeto de conexão mostra mensagens trocadas ou associa elementos de fluxo. Estas ações são representadas por três tipos de componentes: sequência, mensagem e associação. A sequência é um elemento que serve para mostrar a ordem do processo. O fluxo de mensagem, representado por uma seta a tracejado, retrata a comunicação entre elementos. E a associação é retratada por uma linha de pontos e usada para agregar dados, texto e outros artefactos (White, 2004).

Swimlanes

Este elemento constitui todos os intervenientes envolvidos no processo, com o objetivo de organizar melhor o fluxo com os diversos elementos gráficos. Os dois tipos de objetos do *swimlane* são: as *pools*, que representam as diferentes entidades num processo, e as *lanes* que são usadas para dividir as *pools* por departamento ou grupo de pessoas envolvidas nesses processos (White, 2004).

Artefactos

A última categoria são os artefactos, usados para fornecer informações adicionais sobre o processo que não afeta o fluxo (Chinosi & Trombetta, 2012). Estes são constituídos por três tipos: objeto de dados, grupos e notas. Os objetos de dados servem para apresentar os dados que são solicitados ou produzidos durante, ou no fim, dos processos. Os grupos permitem agrupar atividades ou notas de texto que detalhem o fluxo de objetos (White, 2004).

3 Descrição e Caracterização do Caso de Estudo

O presente capítulo tem como objetivo descrever e caracterizar o projeto numa situação real, explicando de uma forma detalhada o processo de gestão de equipas de software. Portanto, a origem deste projeto surgiu da necessidade de melhorar os processos de gestão de equipas de produto SaaS, desde o levantamento de necessidades dos diferentes *stakeholders* associados à empresa, até ao lançamento de uma nova versão do produto. No entanto, primeiramente, é importante perceber mais especificamente a fonte do problema real ao longo deste processo complexo, bem como obter a perceção e conhecimento do funcionamento atual do departamento de desenvolvimento de produto.

Desta forma, o primeiro passo na caracterização do caso de estudo baseia-se no levantamento do modelo *As-Is* da gestão das equipas de produto. A primeira secção retrata como é que o departamento de desenvolvimento está organizado, a segunda fornece uma visão geral dos processos de funcionamento do departamento de produto, na dimensão de desenvolvimento de software, e como eles interagem para cumprir a sua missão e visão. Por fim, na terceira é descrita a metodologia de desenvolvimento usada.

3.1 Estrutura do Departamento de Desenvolvimento de Produto

A empresa possui o controlo total da cadeia de valor, sendo constituída pelos diversos departamentos necessários para o seu funcionamento. Na Figura 3 está detalhado a estrutura do departamento de Desenvolvimento de Produto da E-goi, que é responsabilidade do CTO.

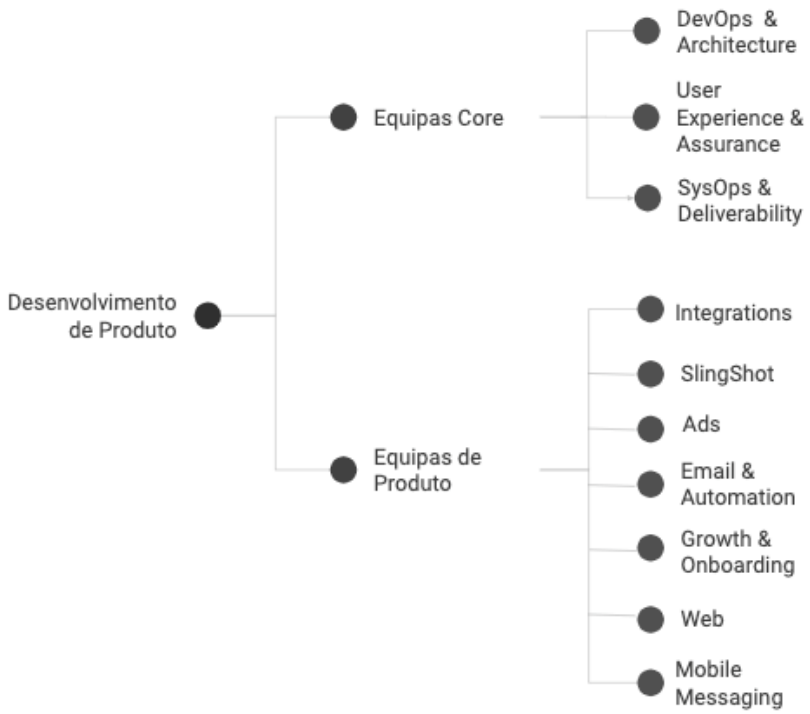


Figura 3 - Estrutura do departamento de desenvolvimento de produto da E-goi

Como podemos verificar, este departamento é subdividido em duas equipas: Equipas Core e Equipas de Produto. As Equipas Core são especializadas em determinadas disciplinas, envolvidas no desenvolvimento de produtos tecnológicos com o papel de mediação, apoio e gestão macro do ecossistema da empresa. Estas são compostas por três equipas, *DevOps & Architecture, User Experience & Assurance* e *SysOps & Deliverability*.

As Equipas de Produto são multidisciplinares e responsáveis pela estratégia, roadmap e definição de funcionalidades de um produto ou linha de produto. Existem sete que estão distribuídas pelas diferentes componentes - *Integrations, SlingShot, Ads, Email & Automation, Growth & Onboarding, Web, Mobile Messaging* -, constituídas pela equipa de produto formada por 2 a 3 elementos (1 *lead developer*), e pelo *Product Manager*, que é o responsável por geri-la.

3.1.1 Descrição das Equipas de Produto

Integrations

Esta equipa tem o intuito de aumentar o valor das integrações no ecossistema da empresa. A integração facilita as empresas que pretendem sincronizar a base de dados da sua loja integrada com a E-goi. Portanto, as principais responsabilidades e áreas de atuação desta equipa de produto incluem:

(i) Realizar parcerias de integração;
(ii) Desenvolvimento de *plugins* de integração com a plataforma e-goi:
(iii) Manutenção corretiva e evolutiva de API V2;
(iv) Desenvolvimento da API V3.

SlingShot

O principal objetivo desta equipa de produto de software é aumentar o número de mensagens transacionais enviadas. As mensagens transacionais acontecem no seguimento de uma determinada ação. Portanto, as responsabilidades e áreas de atuação deste componente de produto são as seguintes:

(i) Comunicações transacionais através dos canais de comunicação existentes (SMS e E-Mail);
(ii) Relatórios de envios transacionais;
(iii) Mensagens registadas (mensagens certificadas com evidência);
(iv) Facilitar a integração de mensagens transacionais.

Ads

O *Ads* é um componente do produto de software, que possibilita aos utilizadores captarem o maior número de clientes que ainda não constam na sua base de dados. Para isto ser possível, a plataforma permite criar anúncios simples e eficazes nos principais sites de pesquisa e nas maiores redes sociais de uma vez só. Adicionalmente, os dados do anúncio podem ser acompanhados por meio de um relatório único no qual consta informação essencial para avaliar o sucesso nos diversos canais publicados. Assim sendo, esta equipa tem as seguintes responsabilidades e áreas de atuação:

(i) Impulsionamento de campanhas de email, *landing pages* e produtos;
(ii) Distribuição para vários canais de comunicação globais e locais;
(iii) Simplificar o processo de criação e relatório de anúncios.

Email & Automation

Esta equipa é responsável por aumentar o uso do e-mail e mensagens automatizadas (*Autobots*). Esta componente permite tornar automático o ciclo de vendas do E-Commerce ou visitas no blogue dos clientes. Em função disso, é possível interagir com os clientes nos canais disponíveis, baseado nos seus perfis e comportamentos e enviar as mensagens apropriadas, no momento certo para fidelizá-los. Em termos de responsabilidades e áreas de atuação, a equipa de *Email & Automation* abrange os seguintes pontos:

(i) Manutenção corretiva evolutiva;
(ii) Desenvolvimento dos componentes de maior utilização e impacto na plataforma;
(iii) E-mail e automação (*Autobots*).

Growth & Onboarding

O principal objetivo desta equipa é melhorar a experiência e aumentar o volume de negócios ao longo do processo de compra e integração. Esta componente da E-Goi está relacionada com os elementos de grande utilização inicial, como as listas de contactos dos clientes (base de dados) e remetentes. Por outro lado, esta equipa também se encarrega pela experiência e conversão do cliente, incluindo os pagamentos, modelo de negócio, checkout, funil de resultados e vendas. Portanto, as suas responsabilidades e áreas de atuação englobam:

(i) Componentes diretamente relacionados com a experiência e conversão;
(ii) Componentes com grande utilização na experiência inicial;
(iii) Desenvolvimento do novo modelo de negócio.

Web

O foco da equipa *Web* é aumentar a base de dados e o envolvimento dos clientes. Esta parte do produto, de modo a atingir o objetivo, é constituída por várias ferramentas que auxiliam o utilizador a captar novos clientes, a criar conteúdo relevante para transformar os visitantes em compradores e a recuperar clientes inativos. Logo, os compromissos e atividades que esta equipa possui são baseados no desenvolvimento e manutenção corretiva das seguintes funcionalidades:

(i) Formulários e janelas *Popups*;
(ii) *Landing Pages*;
(iii) Notificações Web;
(iv) Campanhas de abandono de carrinho.

Mobile Messaging

Anteriormente era a equipa de *Ads* que geria as funções a serem desempenhadas. De momento as suas responsabilidades e áreas de atuação estão em desenvolvimento e subdivididas nas seguintes componentes / linhas de produto:

(i) SMS e *Smart* SMS;
(ii) Transmissão de voz;
(iii) Notificações de *Push* no telemóvel.

3.1.2 Cargos Operacionais e Respetivas Funções

As funções de cada elemento serão apresentadas, detalhadamente, com o objetivo de elucidar o papel de todos os intervenientes no processo de desenvolvimento de produto de software, identificando quem os executa e definindo o tipo de responsabilidade que lhes é atribuída (Cf. Tabela 1).

Tabela 1 – Cargos operacionais e respetivas funções

Cargo(s)	**Descrição da responsabilidade**
CEO	É responsável por criar a visão anual da empresa e definir a estratégia e os objetivos de acordo com a missão, produto, metas e as necessidades operacionais da organização para originar lucro.
CTO / *Head of Product*	É encarregado das necessidades tecnológicas, bem como a sua pesquisa e desenvolvimento. Quanto ao *Head of Product*, é encarregue por definir a visão do produto, organizar as equipas multifuncionais e estabelecer a cultura do departamento. No caso da E-goi, tanto o CTO como o *Head of Product* são cargos exercidos pela mesma pessoa.
Product Manager (PM)	É responsável por todas as componentes de uma área de negócio, garantindo a entrega do produto de acordo com o que os *stakeholders* desejam. Além disso, são responsáveis por gerir todos os aspetos relativos a uma determinada equipa – recursos, pessoas, projetos, entre outros – e fornecer orientação, garantindo a satisfação dos objetivos.
Product Team (PT)	Têm como foco desenvolver produtos que ofereçam soluções aos utilizadores e pela manutenção corretiva. Para além disso são responsáveis, durante o planeamento, por analisar, com o seu PM, o esforço estimado das tarefas a realizar.
Quality Assurance (QA)	É uma equipa core, responsável por garantir que o produto de software esteja de acordo com a qualidade exigida, ou seja, evita que o produto chegue ao cliente com erros ou problemas de funcionamento.
User Experience (UX)	Fazem parte das equipas core, e são responsáveis por garantir a melhor experiência ao utilizador. Isto envolve o design de todo o processo de aquisição e integração do produto, incluindo aspetos de *branding*, design, usabilidade e função. Adicionalmente, esta equipa atualiza os conteúdos educativos para comunicar externamente as novas funcionalidades lançadas no produto.
DevOps & Architecture (RM)	É uma equipa core, constituída por um elemento, responsável pelo ciclo de vida do lançamento, focando-se na coordenação de vários aspetos da produção e dos projetos numa solução integrada. Além disso, é responsável por guiar tecnicamente as equipas de produto, através do seu conhecimento de programação e boas práticas, e dos problemas e limitações existentes.

Masters	Os Masters na E-goi são todos os gestores das equipas existentes no departamento de desenvolvimento de produto, englobando os das equipas core e equipas de produto.
Head of Department (HD) / Stakeholders internos	Os gestores de departamento, durante o desenvolvimento de produto, são responsáveis por revelar as suas necessidades. Estes, em conjunto com os restantes membros da empresa, são os *stakeholders* internos, que colaboram com o departamento de desenvolvimento de produto para reportar *issues* relativos ao sistema.

3.2 Análise dos Processos de Gestão de Equipas de Produto

A documentação existente e as entrevistas informais com os principais *stakeholders*, nomeadamente os proprietários dos processos e a observação dos colaboradores no cumprimento das suas tarefas e processos diários, permitiram recolher informação essencial para desenvolver o mapa que representa a presente cadeia de valor da gestão das equipas de produto de *software* da empresa (Cf. Figura 4).

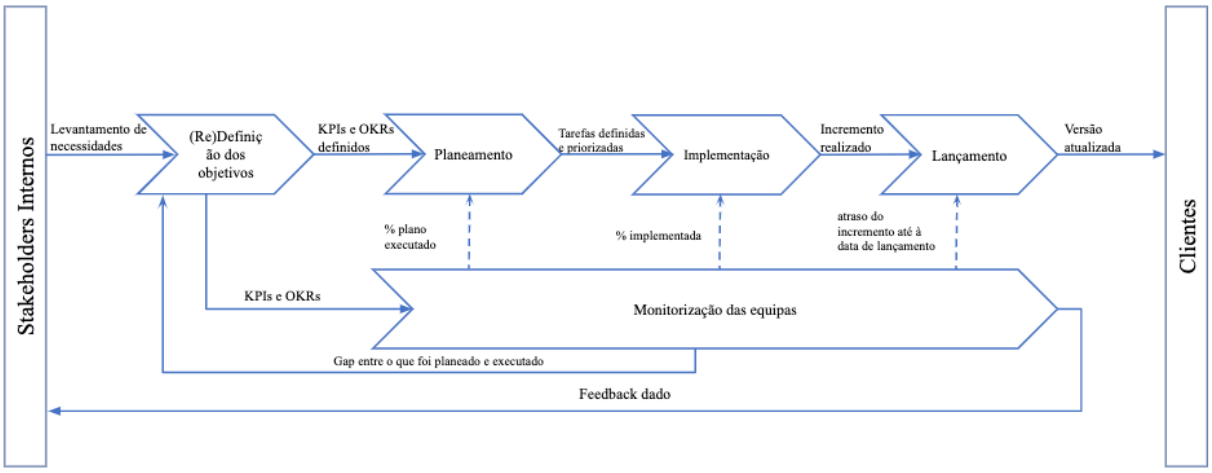


Figura 4 - Mapa de Processos As-Is

Seguidamente, encontra-se uma breve descrição dos cinco processos críticos identificados na análise do departamento do produto na E-goi:

(i) (Re)Definição dos objetivos: O propósito deste processo consiste na definição do que vai ser desenvolvido durante o trimestre, conforme a estratégia traçada pela empresa. Com base no levantamento de necessidades dos diversos departamentos (*stakeholders*), são definidos os Objetivos de Resultados Chave (OKR) para cada equipa, com o intuito dos colaboradores do departamento de desenvolvimento de produto saberem quais as metas a cumprir, para a empresa alcançar os objetivos estratégicos delineados. Se após a monitorização mensal dos OKR's e KPI's se verificar que o progresso não está de acordo com o que estava planeado, os objetivos são ajustados mediante o que é prioritário cumprir.

(ii) Planeamento: Este processo é executado semanalmente, estando presente o CTO e os Masters, para discutir o que foi realizado na semana anterior, os planos para a semana seguinte, questões de necessidade de colaboração entre equipas e as dificuldades em

geral. Adicionalmente, o CTO analisa o que foi discutido por cada equipa e sugere soluções para as questões mais prioritárias.

(iii) Implementação: Este processo visa o desenvolvimento de software, de forma a atender aos objetivos estabelecidos no início do trimestre. Com base no planeamento semanal, as necessidades dos *stakeholders* encontram-se devidamente priorizadas para serem desenvolvidas diariamente. Este é dividido em diversas fases, incluindo a criação da *mockup*, o design da funcionalidade, a implementação das tarefas e os respetivos testes.

(iv) Lançamento: Este processo consiste na análise e integração dos incrementos realizados pelas várias equipas de produto. Após a análise cuidada do código-fonte, é realizado o lançamento da nova versão, pela qual é comunicada internamente através da plataforma de colaboração da empresa, *Bitrix24*. Caso a mudança seja relevante, é também divulgado aos clientes nos conteúdos de educação usados pela E-Goi (Plataforma, Blogue,Knowledge Base, curso Udemy, Newsletter).

(v) Monitorização das equipas: Este processo é transversal aos processos de planeamento, implementação e lançamento, de forma a ser realizada uma análise mensal de ocorrências e imprevistos. Esta análise é feita com base no acompanhamento de KPI e OKR que estão atribuídos a cada equipa do departamento de desenvolvimento de produto. Após a análise, se necessário, os objetivos são ajustados de forma a ser possível o cumprimento dos objetivos mais relevantes para a empresa.

Tendo por base a lista de cargos operacionais elaborada no subcapítulo 3.1.2, as fases e as responsabilidades de cada processo estão representadas através da matriz RACI no *Anexo B*. De seguida, serão descritas com um maior detalhe as diversas fases representadas, de forma a entender-se o fluxo de trabalho e a interação existente entre os diferentes intervenientes de cada processo, especificados nos *Anexos C, D, E, F* e *G*, através de *swimlanes* e fluxogramas.

3.2.1 Processo da (Re)Definição de Objetivos

O processo da definição de objetivos, realizado trimestralmente, serve para definir o que será desenvolvido pelas diversas equipas do departamento de desenvolvimento de produto. Se este, por algum motivo, for feito incorretamente ou desorganizadamente, todo o restante processo poderá ficar comprometido.

Este processo inicia-se com o levantamento de necessidades por parte de cada gestor de departamento. Anteriormente, com base nas necessidades levantadas, o CTO reunia-se com todos os gestores de departamento durante um dia, para realizar uma listagem com a descrição dessas necessidades, para de seguida ser apresentada aos gestores das equipas core e de produto. Esta listagem encontrava-se descentralizada e dispersa pela não integração da mesma num documento único, sendo que existiam necessidades que não iriam ser satisfeitas no trimestre que estava a ser planeado e, consequentemente, alguma informação acabava por ser esquecida.

De seguida, o CTO estabelecia uma data, para cada Master analisar as necessidades. Deste modo equilibravam os objetivos a serem cumpridos no trimestre e realizavam uma proposta de OKR, que estariam associados aos KPI, para obterem um maior controlo sobre os objetivos. Este processo costumava ser demorado – problema identificado pelos PM -, devido a muitas necessidades serem vagas e necessitarem de uma melhor explicação por parte dos *stakeholders* internos.

Com base na proposta do *roadmap* planeado por cada equipa, o CTO marcava uma reunião com todos os Masters para analisar e discutir as suas propostas e para decidir o que seria mais prioritário satisfazer. Posteriormente, o CTO apresentava a lista de propostas ao CEO, de forma a alinhar os objetivos conforme a estratégia delineada pela empresa. Com fundamento no que foi discutido, o CTO ajustava os objetivos e comunicava aos gestores das equipas core e de produto. Seguidamente, cada Master tinha a responsabilidade de organizar os seus objetivos, através da repartição dos mesmos em tarefas. Assim que estas estivessem devidamente explícitas, o gestor comunicava à sua equipa de produto e colocava-as no JIRA - software utilizado para gerir e organizar as tarefas de todo o departamento de desenvolvimento de produto.

Durante o trimestre, havendo um controlo mensal do progresso dos objetivos de cada equipa – processo de monitorização das equipas -, e dependendo da sua performance, o plano estabelecido e acima descrito, poderá estar sujeito a uma redefinição.

3.2.2 Processo de Planeamento

O processo de planeamento é realizado semanalmente e é destacado por três subprocessos principais: priorização de necessidades, descrição do que foi realizado e definição do que vai ser feito. A priorização de necessidades é responsabilidade dos gestores das equipas core e das equipas de produto. Este subprocesso passa por analisar as diversas *issues* criadas pelos membros da empresa e categorizar o *scoring*, de forma a desenvolver o que é mais prioritário e ao que vai encontro dos objetivos estabelecidos.

A categorização do *scoring* é realizada na perspetiva do cliente e do produto. Relativamente à perspetiva do cliente existe *Customer Business Value* (medida relacionada com o número de segmentos que afeta), *Customer Impact* (medida relacionada com o número de pessoas, que constam no público alvo, que serão impactadas), *Customer Satisfaction* (quão maior o número de segmentos satisfeitos, menor a prioridade). No que respeita ao *scoring* na perspetiva do produto, são usados três parâmetros para considerar a prioridade: *Product Reach* (medida que reflete o número potencial de utilizadores afetados), *Product Impact* (medida que descreve quão grande será o impacto em cada utilizador)*; Product Effort* (consoante o tempo, é avaliado o custo, quanto maior o custo, menor é a prioridade).

Posteriormente à realização do scoring de ambas as perspetivas, é realizado o seguinte cálculo para o scoring final: (perspetiva do cliente + perspetiva do produto) / 30*10. Consoante o resultado final, é aplicado o método MoSCoW, consistindo num método de priorização usado para decidir quais os requisitos que devem ser desenvolvidos primeiro, e os que podem ser excluídos. Existem as seguintes quatro categorias:

(i) *Must-have* (8-10): Se o produto não funcionar sem o requisito;

(ii) *Should-have* (6-8): a *issue* tem um elevado valor de negócio, mas o sistema pode funcionar sem ele, embora, se possível, seja incluído;

(iii) *Could-have* (4-6): Este requisito não tem grande impacto no valor de negócio, mas ainda assim tem valor, tendo um impacto muito menor se for deixado de fora;

(iv) *Will not have* (2-4): é considerado uma *issue* com o valor mais baixo, por isso não será incluído no momento, de modo a poupar energia e tempo para realizar as *issues* com maior prioridade.

Considerando a importância da necessidade que é avaliada conforme o *scoring* final, os gestores das equipas core, e/ou de produto, decidem se vão corrigir ou implementar, dependendo do tipo de *issue*, na semana seguinte. De ambas as formas, cada *issue* tem que ser associada a um tipo de resolução, independentemente se vai ou não ser solucionada, de modo a reportar ao responsável pela sua criação que esta foi devidamente tratada. Portanto, depois das necessidades estarem devidamente definidas, são distribuídas por cada elemento de equipa tarefas, atentando o tempo de esforço estimado. Com o plano já definido e pronto para implementar, o próximo subprocesso visa a comunicação entre as diversas equipas, de forma a controlar o que foi implementado e o que vai ser, conforme o plano executado. Consequentemente, existe uma reunião todas as sextas-feiras com todos os *Masters*, onde se discute o que foi realizado durante a semana e o que vai ser feito na próxima, tendo uma duração total de 1 hora. Nestas reuniões é apresentado por cada gestor o resumo da semana anterior, o plano para a próxima e referindo, se necessário, a necessidade de colaboração entre as equipas. Para além disso, são abordadas as dificuldades que cada equipa obteve e discutidas possíveis soluções para as ultrapassar.

3.2.3 Processo de Implementação

No caso de ser necessário desenvolver uma funcionalidade nova, o fluxo normal do processo de implementação inicia-se com a criação da *mockup*. Esta visa exibir o esqueleto da nova funcionalidade e dar instruções de como a interface deve ser representada. Assim que a equipa de *UX* tem acesso a esta *mockup*, fica responsável por realizar o design requerido pelo PM e quando estiver validado, notifica a equipa responsável.

De seguida, a equipa de produto implementa a funcionalidade ou corrige os *bugs*, conforme o tipo de *issue*. Neste processo, a função do PM passa por acompanhar a sua equipa, com o objetivo de esclarecer dúvidas e evitar o envio da *issue* sem estar testada, para o *QA*. No caso de surgirem imprevistos ou dificuldades durante a implementação da tarefa, que possa bloquear o fluxo, o CTO é consultado para auxiliar na sua resolução.

Assim que a *issue* estiver implementada e testada pela equipa de produto, é enviada através do *JIRA* para a equipa de *QA* para ser novamente analisada e testada antes de seguir para produção. No entanto, como existem sete equipas de produto, a equipa de *QA* tem de gerir os testes a serem realizados, mediante o *scoring* de cada funcionalidade. Estes são somente testes funcionais, que consistem na validação do sistema de software em relação aos requisitos funcionais sem verificação do código-fonte. Portanto, são realizados testes de iteração para certificar a inexistência de erros. Na hipótese de os requisitos estarem a funcionar incorretamente, ou estarem incompletos, são explicados e enviados novamente através do *JIRA* à respetiva equipa para serem corrigidos. Por outro lado, se o comportamento da funcionalidade for o esperado, esta será enviada para produção e seguidamente lançada para a próxima versão do produto.

No momento em que a respetiva funcionalidade estiver disponível para todos os utilizadores, o *QA* analisa o comportamento dos clientes via *Full Story*, que consiste num software que permite capturar dados de interação dos utilizadores com a plataforma, em tempo real.

3.2.4 Processo de Lançamento

O processo de lançamento é composto por três fases principais: pré-lançamento, lançamento técnico, e o pós-lançamento. O pré-lançamento consiste na definição dos épicos –

funcionalidades maiores – e/ou *commits* – alterações mais recentes efetuadas do código fonte no repositório – por parte do *PM*, que anteriormente tinha de os reportar ao *Release Manager / Tech Lead*, através do chat interno. O *PM*, para informar o *RM* sobre os épicos e *commits* que iriam entrar em produção, tinha que se reunir com o seu *Lead Developer*, originando processos desnecessários – problema encontrado pelos *PM* -, enquanto poderia ser só o *Lead Developer* a ficar encarregue deste processo, devido a ter mais conhecimento do estado de desenvolvimento.

O lançamento técnico é realizado todas as segundas-feiras e/ou quartas-feiras, sendo responsabilidade do *Release Manager*. O primeiro passo é preparar o *merge request*– ferramenta de revisão de código que serve para juntar todos os projetos -, no caso de ser necessário remover algo da *bug tracking* – servidores de testes para onde a maioria das *issues* de correções são testadas –, é criada uma *branch* separada da *release* e feita a integração de todos os *commits* reportados. De forma a distinguir aas diferentes versões, são colocadas etiquetas e, de seguida, analisados e testados os *commits* para verificar se o código está bem implementado, e se não existem linhas de código a mais que possam diminuir a performance e a estabilidade do sistema. Na eventualidade de uma determinada *branch* conter erros, o *Release Manager* colabora com a respetiva equipa de produto para auxiliar na parte técnica. Dependendo do impacto e da gravidade do problema, a junção da *branch* à *branch master* - branch principal na qual existe somente versões que estão em produção - pode ser adiada, de forma a evitar problemas na futura versão. Por outro lado, se forem erros triviais, são corrigidos no momento para que, de seguida, seja realizada a junção de todas as *branches* prontas, à *branch master*.

A fase pós lançamento consiste na comunicação realizada internamente e/ou externamente. Assim que a nova versão estiver em produção, é atualizado no JIRA e as *issues* que foram resolvidas são descritas e publicadas no *Bitrix24* - plataforma de comunicação interna da empresa-, e posteriormente explicadas num evento semanal. Os atributos das *issues* publicadas estão representados na Tabela 2.

Tabela 2 - Atributos publicados no *Bitrix24* da nova versão

Issue ID	Issue Type	Application Component	Issue Summary	Issue Resolution

O evento semanal acima mencionado é realizado todas as sextas-feiras, pelo CTO, com o intuito de apresentar as presentes e futuras atualizações a todos os membros da empresa. Quando as alterações implementadas forem relevantes para serem divulgadas ao cliente, a equipa de UX fica responsável por fazê-lo nos diversos canais de comunicação da E-Goi (Plataforma, Blogue, Knowledge Base, curso Udemy, Newsletter).

3.2.5 Monitorização das Equipas de Produto

Embora a monitorização não seja um processo tão frequente quanto a implementação, sendo que é realizado mensalmente, exige muito envolvimento por parte das equipas de produto e dos seus gestores. É um processo transversal ao planeamento, implementação e lançamento, cujo foco é analisar as atividades desenvolvidas pelas equipas do departamento de desenvolvimento de produto, de forma a averiguar se os objetivos estão a ser cumpridos conforme foram estabelecidos previamente.

Este processo recai essencialmente sobre os *Masters*, que têm a responsabilidade de controlar e monitorizar os KPIs e OKRs da sua equipa. As métricas referentes a cada componente são atualizados através de um *script*, de forma a controlarem tudo em tempo real. Os objetivos são divididos em resultados-chave e, ao contrário das métricas, têm de ser atualizados manualmente por cada gestor à medida que vão sendo cumpridos.

De acordo com o progresso de cada equipa, o CTO, em conjunto com o *Master*, concentra-se em analisar se o projeto está a progredir de acordo com o plano e o cronograma estabelecidos no início do trimestre. Esta análise serve para identificar as ocorrências e imprevistos do projeto e, se necessário, ajustar o planeamento considerando o que é mais urgente realizar. Após esta monitorização mensal, as eventuais alterações realizadas no plano são comunicadas aos *stakeholders* internos.

3.3 Metodologia de trabalho da empresa

De forma a haver uma gestão visual do que é necessário desenvolver, a empresa utiliza a metodologia de desenvolvimento Kanban, que se rege por princípios que estão descritos no subcapítulo 2.2.1. Esta gestão é feita através do JIRA, que é o software utilizado para gerir e organizar as tarefas de todo o departamento de desenvolvimento de produto.

3.3.1 Criação de Issues

A criação de *issues* pode ser realizada por todos os membros da empresa, que usem diariamente a plataforma E-Goi, de forma a auxiliar o departamento de desenvolvimento a detetarem erros existentes e sugerirem melhorias ou novas funcionalidades. No entanto, visto que o JIRA tem um número limite de utilizadores e se destina para a gestão do departamento de desenvolvimento de produto, impossibilitando a utilização por parte dos restantes departamentos, foi criada uma nova funcionalidade - *Issue Creator* – na plataforma Admin desenvolvida pela empresa, e que está integrada com o JIRA. A *Issue Creator* permitia selecionar o tipo de *issue* – *bug,improvement, new feature, task* -, a prioridade considerando o modelo *scoring*, a equipa responsável por resolver a *issue*, um resumo e uma descrição detalhada da *issue*. Contudo, considerando que cada tipo de *issue* tinha um tratamento diferente, e eram criadas da mesma forma – problema identificado pelos PM -, por vezes havia uma falta de coerência na sua criação, originando má interpretação devido à informação passada ser ambígua e/ou incompleta. Visto que a análise e a seleção das *issues* a serem resolvidas são efetuadas no processo de planeamento, estes problemas têm impacto originando atrasos durante este processo.

3.3.2 Workflow do quadro Kanban

O quadro Kanban customizado pela empresa está dividido em oito colunas, que consistem nos estados em que uma *issue* pode ultrapassar, sendo a sua totalidade o funcionamento do fluxo de produção do departamento de desenvolvimento. A transparência do progresso dos estados é fundamental para o bom funcionamento da empresa, devido a permitir uma maior visibilidade aos *Masters* do trabalho que falta executar e o que foi executado.

Tal como o progresso dos estados, também é essencial, após a *issue* ser devidamente analisada e/ou desenvolvida, atribuir um tipo de resolução para que quem a abra saiba que o problema que levantou já foi analisado. A comunicação é feita para toda a empresa pelo CTO, no pós-lançamento, através da publicação de uma nova versão, como referido na subsecção 3.2.4.

Na Figura 5 está ilustrada a organização do quadro de operações Kanban. Existem dois quadros, sendo que o superior é designado para resolução de *issues* e o inferior para os épicos.

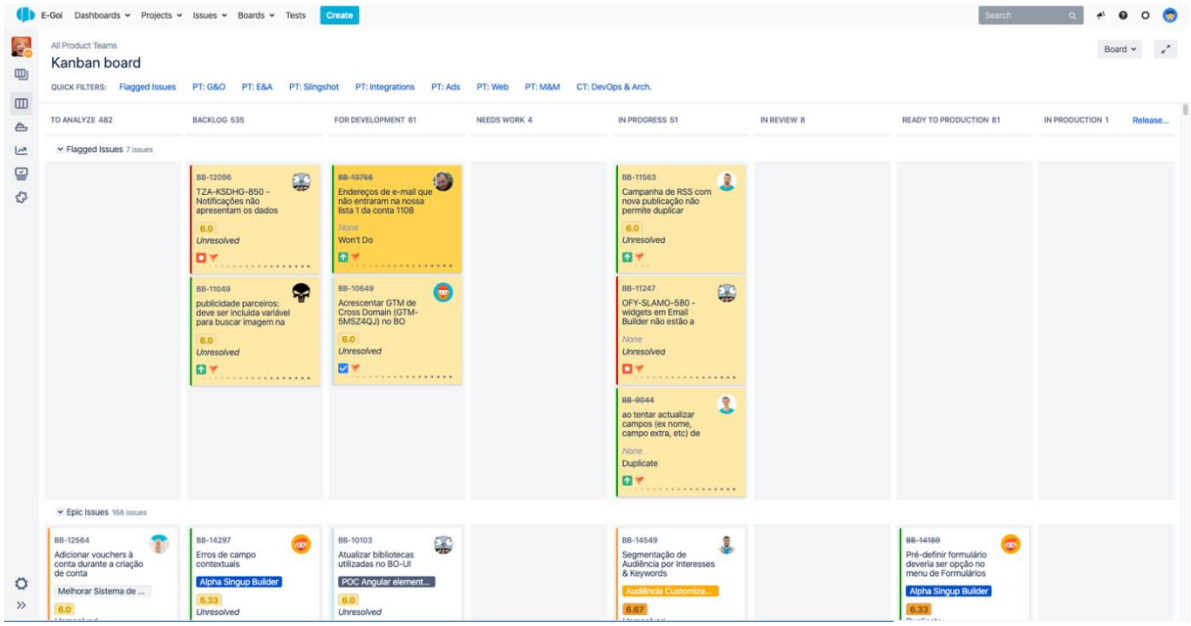


Figura 5 - Quadro Kanban da E-goi

Como referido anteriormente, todos os funcionários da E-goi podem criar *issues* e todas aparecerão na primeira coluna da equipa responsável por resolver. A análise das diversas tarefas é feita com base na avaliação de cada PM. No *Backlog* (segunda coluna) já se encontram as tarefas que foram analisadas e que vão ser desenvolvidas. As tarefas que têm maior prioridade, com base no modelo de priorização descrito no processo de planeamento, são as que devem ser inseridas na coluna "For Development" (terceira coluna). Por outras palavras, todas as funcionalidades que estão planeadas a ser desenvolvidas na semana seguinte encontram-se nesta coluna. No momento em que o responsável da equipa de produto inicia o desenvolvimento de uma determinada tarefa, arrasta o cartão correspondente para o estado "In Progress", sendo que o responsável da abertura da *issue* é notificado via email. Assim que o responsável pela tarefa entende que a funcionalidade já está corrigida ou implementada, arrasta para a coluna "In Review". Nesta coluna, a equipa QA é responsável por analisar e testar as diversas tarefas desenvolvidas pelas equipas de produto. No caso de identificarem algum problema, têm de descrever o que está a falhar e colocar o cartão na coluna "Needs Work", para ser devidamente corrigido. Por outro lado, se após o teste de qualidade o QA aprovar a *issue*, arrasta o cartão para "Ready to Production", no qual o RM analisa todos os *commits* e épicos que estão prontos para produção e realiza o lançamento da nova versão do produto.

4 Identificação e Caracterização do problema

Na sequência do levantamento dos processos de gestão de equipas de software da empresa, este capítulo é dedicado à identificação das oportunidades de melhoria, com o intuito de responder à *QP1*.

Considerando a cultura organizacional, não existem dados que permitam a uma nova equipa/gestor entender ao que é suposto responder e, mesmo existindo, não subsiste nenhum reflexo da mesma que seja facilmente entendida e integrada por todos os colaboradores do departamento de desenvolvimento de produto. Assim, existem problemas que urgem resolver porque têm impacto em outros processos, nomeadamente o de definição de objetivos, planeamento e lançamento. Uma vez que a empresa cresceu rapidamente, o departamento passou por um processo de transformação do modelo organizacional, causando o desalinhamento de práticas entre equipas, retrabalho e falta de normalização e sistematização de alguns processos.

A identificação e procura destes problemas teve como base as metodologias e princípios referidos ao longo do capítulo 2. Adicionalmente, e como mencionado anteriormente, a pesquisa qualitativa serviu de abordagem ao caso de estudo, através da realização de entrevistas, observação de alguns participantes no desempenho das suas atividades e com base no modelo *As-Is* dos processos de gestão de equipas de software. Portanto, tanto a revisão literária como a informação adquirida através do levantamento de processos permitiram a deteção dos problemas apresentados nos seguintes subcapítulos.

4.1 Problemas no Processo de Definição dos Objetivos

De forma a compreender-se o funcionamento do processo de definição de objetivos, bem como os gargalos existentes, realizou-se uma análise do processo. Com base nas entrevistas, constatou-se que o principal problema deste processo era o atraso que acontecia todos os trimestres devido à pouca maturidade do mesmo.

P1: Atraso do início do trimestre

O processo da definição dos objetivos é transversal e montante a todas as equipas do departamento de desenvolvimento de produto. A finalidade deste processo é recolher as necessidades de todos os departamentos da E-goi, para serem enquadrados nos OKR que irão ser o foco ao longo do trimestre. Além disso, é um processo extremamente decisivo, o que torna crucial dinamizar o máximo possível, para que o trimestre seja planeado com sucesso.

Como a definição dos objetivos do trimestre seguinte era somente realizada entre o fim do trimestre anterior e o início do seguinte, atrasava em média 15 dias, desperdiçando tempo de trabalho. O ideal seria que os objetivos estivessem estabelecidos antes do trimestre começar evitando a acumulação de trabalho e tornando o fluxo mais fluído.

> *"Os objetivos do próximo trimestre são definidos entre o fim do trimestre e o início do próximo. Sendo que o problema deste processo é desperdiçarmos 15 dias do trimestre só a definir objetivos, quando poderíamos estar a desenvolvê-los. Consequentemente afeta todo o departamento de desenvolvimento de produto, porque depois todas as equipas têm uma maior pressão para desenvolver rapidamente mockups, interfaces, comunicar com a sua equipa, etc. Se fossem definidos mais cedo, seria o ideal."* – PM da equipa *Email & Automation*

- (C1.1) Especificação de necessidades pouco percetível e unidirecional

Apesar de existir o processo de definição de objetivos, a maturidade era baixa devido a não existir uma regularização concreta que permitisse a todas as equipas do departamento de desenvolvimento uma melhor percetibilidade das necessidades e das prioridades dos *stakeholders* internos. Portanto, este pouco detalhe e clareza das necessidades originava dúvidas nas diferentes equipas do departamento de desenvolvimento, dificultando a realização da proposta de OKR e, consequentemente, desperdiçava-se tempo no esclarecimento das mesmas com os respetivos *stakeholders*. Adicionalmente, não existia forma sistemática de coletar e cruzar necessidades dos departamentos existentes na empresa, sendo que o levantamento era unicamente direcionado às necessidades dos *stakeholders* internos para o departamento de desenvolvimento.

> *"Falta um maior envolvimento dos outros departamentos... deveriam envolver-se mais para as decisões estratégicas."* – HD do Departamento de vendas PME

- (C1.2) Processo de levantamento de necessidades não uniformizado

Como o processo não era uniformizado, cada trimestre era criado um documento onde se escrevia um conjunto de critérios. Como não existia nenhuma *framework* que facilitasse este processo, por vezes a especificação ficava incompleta e/ou impercetível. Por isso, foi deparada a falta de uma ferramenta de suporte que auxiliasse a comunicação e que permitisse sistematizar as necessidades, para que, subsequentemente, cada equipa propor os seus próprios OKR's conforme a estratégia global da empresa.

O problema e as causas identificadas no processo Definição dos Objetivos encontram-se sintetizadas na Tabela 3.

Tabela 3 - Resumo dos problemas identificados no processo de Definição dos Objetivos

P1	Atraso do início do trimestre	
	C1.1.	Especificação de necessidades pouco percetível e unidirecional
	C1.2.	Processo de levantamento de necessidades não uniformizado

4.2 Problemas no Processo de Planeamento

Derivado à falta de sistematização que existia no *Issue Creator*, e sendo este usado por todos os colaboradores da empresa, era visível uma falta de normalização na criação das *issues*. Além disso, foi identificada a falta de documentação de *issues* no JIRA que complementasse os estados e as *issues resolutions* existentes.

P2: Falta de Normalização na Criação de *Issues*

Sendo que a criação de *issues* é feita por qualquer pessoa da empresa, e visto que não existia uma normalização da informação que deveria ser passada na caracterização das *issues*, foi necessário melhorar esta ferramenta. Esta caracterização é essencial para corrigir ou implementar funcionalidades do sistema, e é fundamental que seja realizada de uma forma completa e percetível, de modo a que quem vai analisar não necessite de pedir mais informações. No entanto, a forma como esta era criada não estava devidamente sistematizada,

por não haver campos necessários para cada tipo de *issue*. Isto é, como existem 3 tipos de *issues* frequentes – *bugs, improvements, features* –, cada uma delas tem um tipo de tratamento diferente, no entanto todas elas eram criadas da mesma forma.

De acordo com Petersen (2010); Poppendieck and Cusumano (2012) este problema é identificado como um desperdício – movimentos e troca de tarefas – devido a por vezes os *Masters* necessitarem de esclarecer dúvidas sobre as *issues* criadas que originava perdas de concentração e maior dificuldade a retomar as tarefas.

- (C2.1) Falta de uma ferramenta sistematizada que normalize a informação que é passada

Como todos os utilizadores internos da empresa têm permissão para abrir *issues*, seria importante existir uma ferramenta que os permitisse especificar de uma forma coerente e completa os diferentes tipos. Como referido no subcapítulo 3.3.1, os campos existentes na criação de *issues*, através do Admin, são insuficientes devido à sua falta de detalhe, originando problemas que estão descritos na C2.2. Por isso, era evidente a falta de uma ferramenta sistematizada que normalizasse a informação que se pretendia transmitir na abertura de *issues*.

> *"Falta uma coerência e uma normalização no processo de criação de uma issue, devido aos diversos departamentos terem permissão para as criar. Poupava muito tempo ao PM no tratamento de novas issues, se todos os departamentos as criassem de um modo coerente"* - PM da equipa *Email & Automation*

- (C2.2) Especificação das *issues* ambígua e/ou incompleta

A falta da sistematização referido na C2.1 originava problemas como a falta de percetibilidade relativa às *issues* que eram abertas, devido à informação estar incompleta e/ou redundante. Esta pobre caracterização provocava uma má gestão de expectativas, porque em certas ocasiões o que era desenvolvido não correspondia ao que realmente se pretendia. Além disso, de modo a evitar esta má gestão de expectativas e a má interpretação, os *Masters* tinham que esclarecer a *issue* com os responsáveis pela sua abertura. Consequentemente, as equipas de produto tinham muito trabalho a analisar e interpretar as *issues*, correndo o risco de estarem a perder tempo de desenvolvimento.

> *"Por vezes a descrição das issues é ambígua, o que torna complicado perceber o que os stakeholders internos necessitam devido a haver pouco contexto"* - PM da equipa *Growth & Onboarding*

P3: *Issues* Duplicadas

Por vezes a informação que era passada na abertura das *issues* estava duplicada. Este tipo de situações originava trabalho desnecessário em analisar e/ou reproduzir novamente a *issue*.

- (C3.1) Solicitação de feedback pelos vários utilizadores internos da empresa

Como qualquer membro da empresa tem permissão para criar *issues*, existe a possibilidade de mais do que um colaborador encontrar o mesmo *bug* e reportá-lo.

- (C3.2) Não existe validação no momento de criação

Não existindo uma validação que permita aos membros da empresa verificar se a *issue* que pretendem criar já existe, há o risco de sugerirem funcionalidades repetidas, relatarem bugs duplicados e sugerirem melhorias existentes devido a não terem visibilidade sobre as que já se encontram no JIRA.

P4: Documentação das *Issues* incompleta

Após a criação de uma *issue* é necessário haver um tratamento da mesma até estar resolvida.

- (C4.1) Falta do estado "*Require Prior Action*" no fluxo do quadro Kanban

Ao longo deste tratamento existem vários estados pelos quais a *issue* tem de passar, que estão descritos no quadro Kanban criado pela empresa. No entanto, não existe um estado que permita retratar, por exemplo, a falta de recursos e/ou informação ou dependência de outra equipa, impossibilitando o desenvolvimento da *issue*.

- (C4.2) Existem ocasiões sem uma resolução de *issue* apropriada para definir como foi resolvida

De seguida à análise da *issue*, é necessário atribuir uma resolução para que o responsável da sua criação seja informado que esta foi tratada com sucesso. A resolução de uma *issue* é determinada quando o estado da mesma é alterado. Não existindo tipos de resoluções que reflitam uma determinada situação, é importante adicionar *issues resolutions*, que sejam claras para todo o departamento de desenvolvimento e os restantes departamentos da empresa, sendo essencial a sinergia entre eles.

Em resumo, os problemas e as causas identificadas no processo Planeamento estão sistematizadas na Tabela 4.

Tabela 4 - Resumo dos problemas identificados no processo de Planeamento

P2	Criação de *issues*	
	C2.1.	Falta de uma ferramenta sistematizada que normalizasse a informação que era passada
	C2.2.	Especificação das *issues* ambígua e incompleta originando uma má interpretação.
P3	*Issues* duplicadas	
	C3.1	Solicitação de feedback pelos vários utilizadores internos da empresa
	C3.2	Não existe validação no momento de criação
P4	Documentação das issues	
	C4.1	Falta do estado "Require Prior Action" no fluxo do quadro Kanban
	C4.2	Existem ocasiões sem uma resolução de *issue* apropriada para definir como foi resolvida

4.3 Problemas no Processo de Lançamento

Como referido anteriormente, o processo de lançamento é composto por três subprocessos: pelo pré-lançamento, que consiste em reportar os *commits* e/ou os épicos ao RM, pelo lançamento técnico, que engloba a análise das *branches* para posteriormente serem lançadas numa nova versão do produto, e por fim, pelo pós-lançamento, que engloba a comunicação externa e interna das versões. Em todas estas fases foram identificados problemas, que serão retratados ao longo deste subcapítulo.

P5: Falta de conhecimento do PM para reportar as *issues* e/ou épicos

Todas as quintas-feiras, o PM de cada equipa estava encarregue de reportar ao RM os projetos que deveriam entrar em produção (se houvesse alguma modificação), para serem analisados e posteriormente lançados numa nova versão na semana seguinte. Não conhecendo na totalidade o estado do desenvolvimento, o PM tinha que se reunir com a sua equipa de modo a discutirem o que era necessário remover da *bug tracking* para não entrar em produção. Na hipótese de haver épicos prontos para produção, estes também deveriam ser reportados. Com base no primeiro princípio Lean – eliminar desperdícios -, este problema é caracterizado como um processo desnecessário, devido ao PM estar a reunir-se desnecessariamente com a sua equipa, enquanto esta poderia comunicar diretamente com o RM.

> "*O PM por vezes, devido a questões técnicas, tem que esclarecer as perguntas que faço com o seu Lead Developer"* – RM e *Tech Lead* da equipa *DevOps & Architecture.*

P6: Instabilidade do processo de lançamento

Todas as semanas é lançada uma nova versão do produto, como referido no capítulo 3.2.4. Sendo o lançamento técnico total responsabilidade do RM, tem a grande responsabilidade de analisar e corrigir grande parte do código manualmente, sendo que o QA realiza somente testes funcionais. Portanto, as causas de seguida apresentadas são as principais da origem deste problema.

- (C6.1) Falta de recursos

Sendo a equipa de *DevsOps & Architecture* constituída apenas pelo RM/*Tech Lead*, todo o processo de lançamento, incluindo o auxílio técnico às equipas de produto, junção de todas as *branches* reportadas à *branch master*, análise dos *commits*, atualização da base de dados, entre outros, é responsabilidade do mesmo, originando uma sobrecarga de trabalho.

- (C6.2) Falta de testes automatizados

Como a análise das *branches* reportadas são testadas manualmente, são propícios erros humanos, o que torna mais difícil otimizar o código fonte na sua totalidade e gera um consumo de tempo elevado.

A falta de testes automatizados dificulta o trabalho do RM e tem um impacto negativo no sistema devido à possível existência de código não utilizado e ao risco de lançamento de versões instáveis no produto. Consequentemente, o desenvolvimento destes testes aumentaria a performance do sistema, poupando tempo ao RM para se concentrar em outras tarefas. No

entanto, apesar deste tipo de testes consumir menos recursos em execução, o desenvolvimento dos mesmos é mais caro porque exige profissionais especializados nesta área.

> *"Uma coisa que poderia melhorar a qualidade do código e que, consequentemente, garantisse uma maior estabilidade do sistema, seria o desenvolvimento de testes automatizados/unitários, que é algo que falta á empresa."* – RM e *Tech Lead* da equipa *DevOps & Architecture*

P7: Dificuldade em atualizar os diversos canais de comunicação externos, para divulgar as atualizações do produto.

Como existem 7 equipas de produto que todas as semanas incrementam novas alterações no produto – manutenção corretiva, melhoria de funcionalidades, desenvolvimento de novas funcionalidades -, é necessário divulgar aos clientes as respetivas atualizações de forma a entenderem o valor do produto na sua totalidade.

- (C7.1) Falta de recursos

Como só existe um elemento na equipa de usabilidade, que é encarregue por esta comunicação nas diversas fontes existentes em três idiomas diferentes – inglês, espanhol e português –, não existe capacidade suficiente para responder a esta necessidade na sua plenitude. Consequentemente, esta incapacidade acaba por impactar outros departamentos, nomeadamente o comercial e de suporte ao cliente, devido a surgirem mais tickets para o esclarecimento da usabilidade e/ou performance de algumas funcionalidades.

Em resumo, os problemas e causas relacionadas com o processo Lançamento encontram-se sistematizados na Tabela 5.

Tabela 5 - Resumo dos problemas identificados no processo de Lançamento

P5	Falta de conhecimento do PM para reportar as *issues* e/ou épicos	
P6	Instabilidade do processo de lançamento	
	C6.1	Falta de recursos
	C6.2	Falta de testes automatizados
P7	Dificuldade em atualizar os diversos canais de comunicação externos, para divulgar as atualizações do produto.	
	C7.1	Falta de recursos

4.4 Considerações Adicionais

Este subcapítulo é dedicado à identificação de oportunidades de melhoria transversais a todos os processos do departamento de desenvolvimento.

P8: Os colaboradores não sabem como proceder em certas situações

Nas entrevistas realizadas foi verificado que há falta de um processo uniformizado que defina como é que as equipas devem interagir e comunicar no dia-a-dia. Durante o processo de desenvolvimento é essencial que haja interação e comunicação entre as equipas, devido a questões de necessidade de colaboração entre elas e de questões pontuais que possam surgir. Como referido anteriormente, há uma reunião todas as sextas-feiras, na qual estão presentes todos os *Masters* para discutir a necessidade de colaboração entre equipas, no entanto, não

existe uma normalização documental que lhes indique como proceder em determinadas situações diárias. Adicionalmente, existem situações no qual a responsabilidade das *issues* são mal atribuídas, não existindo qualquer tipo de sistematização que indique como é que a *issue* deverá ser comunicada e atribuída ao devido responsável.

- (C8.1) Falta de documentação sobre os processos e fluxo de trabalho

O fluxo existente no departamento é bastante complexo, e com o aumento de número de equipas originou um desalinhamento de práticas entre elas. Portanto, é evidente a falta de documentação disponível sobre os processos e fluxo de trabalho seguidos pelo departamento, sendo especialmente relevante para os novos membros. O não registo destes processos não permitia uma visão global, não existindo, por isso, conhecimento sobre o funcionamento atual do departamento, não sendo aqui cumprido o terceiro princípio *Lean.*

P9: Falta de comunicação entre cliente e equipas de produto

O envolvimento do cliente relaciona-se sobretudo com a comunicação, auxiliando no desenvolvimento de um projeto com sucesso. Este tipo de comunicação entre cliente e equipas de produto na empresa é muito limitada. Hoda et al. (2011) afirmam que a falta de envolvimento do cliente durante o processo de desenvolvimento pode causar vários problemas, nomeadamente na recolha e esclarecimento de requisitos, na insegurança de feedback, perda de produtividade e maior pressão por excesso de comprometimento. No caso da E-goi, das sete equipas de produto, apenas duas têm canal de comunicação direta com o cliente, nomeadamente a equipa *SlingShot* e *Integrations*, devido à API ser pública e serem produtos desenvolvidos à parte. Já as restantes equipas têm contacto indireto, apenas recolhendo feedback fornecido do departamento de vendas e de suporte ao cliente. Com a comunicação indireta, a informação que chega por vezes não é suficiente, ou é mal interpretada por quem a levanta, dificultando a gestão das expectativas dos clientes. Sendo a comunicação existente apenas pós-venda, há uma necessidade de desenvolvimento de um processo que permita o contacto direto do cliente com a empresa.

- (C9.1) Falta de recolha e esclarecimento de requisitos

O levantamento de requisitos é uma fase crucial para o sucesso de um projeto, sendo que é uma condição para um software de qualidade, embora não seja garantia disso. Portanto, o sucesso ou fracasso do desenvolvimento de software depende do cumprimento, ou não, dos requisitos dos utilizadores. De acordo com Carbonell et al. (2009), as interações entre duas partes são consideradas benéficas para a empresa e igualmente importantes para criar valor para o cliente, permitindo partilhar experiências pessoais, influenciar os outros, adquirir competências cognitivas e/ou ajudar no desenvolvimento de novos produtos. No caso da E-goi, sendo esta colaboração inexistente, é originado um crescente custo com a manutenção de sistemas e má interpretação. Esta causa, com base no primeiro princípio Lean, é considerada um desperdício – funcionalidades extra e defeitos – visto não haver uma definição de ambas as partes durante a especificação de requisitos, podendo originar retrabalho.

> *"Por vezes, a equipa de produto tem que perceber melhor o lado do cliente. Isto é, existe uma distância entre aquilo que é feito, e o que o cliente acha que seja mais fácil para ele fazer. Por exemplo, às vezes há uma falha de copy, porque não foi aquilo que realmente*

foi pedido, devido a uma falha de comunicação perante a pessoa responsável." – HD do departamento de Suporte e Apoio ao Cliente

- (C9.2) Risco no desenvolvimento de novas funcionalidades

Sendo o cliente o foco principal da empresa, é necessário satisfazer as suas necessidades através dos produtos e serviços prestados, ouvindo-o desde a abordagem até à pós-venda. Porém, sem o envolvimento do cliente, as funcionalidades são lançadas para o mercado sem uma validação, originando incerteza do seu sucesso e dificuldade de prever as atitudes dos utilizadores. Portanto, a presença do cliente auxilia na avaliação da qualidade em termos de tempo de resposta e qualidade da interface.

"O que por vezes acontece é a dificuldade que existe de as equipas de produto perceberem a necessidade que cliente está a ter." - HD do departamento de vendas Corporativas

Em resumo, os problemas transversais e as causas identificadas estão sistematizadas na Tabela 6.

Tabela 6 - Resumo dos problemas transversais

P8	Comunicação e interação diária entre equipas	
	C8.1.	Falta de documentação sobre os processos e fluxo de trabalho
P9	Não existe colaboração entre as equipas de produto e o cliente	
	C9.1	Falta de recolha e esclarecimento de requisitos
	C9.2	Risco no desenvolvimento de novas funcionalidades

5 Identificação e Desenho de Soluções

As propostas das soluções nos seguintes subtópicos têm como fundamento responder aos problemas identificados na análise realizada do modelo *As-Is* (capítulo 3), pelas necessidades delegadas pelos diversos gestores das equipas core e de produto, e pelo CTO, descritas no capítulo 5, dando assim resposta à QP2.

A realização de workshops para discutir ideias com os participantes do processo foi fundamental para o desenvolvimento soluções para os problemas identificados. Além disso, dado o tempo limitado para o desenvolvimento do projeto e os recursos internos disponíveis, nem todas as soluções foram implementadas.

5.1 Solução para o processo de Definição dos Objetivos

Com o objetivo de abordar as oportunidades de melhoria identificadas no capítulo 4.1, foi desenvolvida a seguinte proposta de ação:

(A1) Matriz origem-destino

Como referido anteriormente, a definição dos objetivos é um processo crucial para o sucesso do trimestre e existia a necessidade de minimizar o atraso que se sucedia no início do mesmo.

Posto isto, foi desenvolvida uma matriz para auxiliar o levantamento de necessidades e consequentemente agilizar o processo. Nas linhas e colunas foram inseridos todos os departamentos da empresa e as equipas do departamento de desenvolvimento de produto. Anteriormente a esta prova de conceito, as necessidades levantadas eram somente dos *stakeholders* internos para o departamento de desenvolvimento, sendo que agora esta matriz possibilita o cruzamento de necessidades entrez os diferentes departamentos e equipas do departamento de desenvolvimento de produto, sendo a data limite do seu preenchimento 15 dias antes ao início do trimestre subsequente. De forma a organizar as necessidades, cada responsável tem de definir o grau de importância de cada uma para o seu departamento. No entanto, cabe aos gestores de cada equipa/departamento realizar a avaliação final de cada necessidade que lhe é requerida. Portanto, para categorizar cada carência é aplicado o método de *MoSCoW* descrito no subcapítulo 3.2.2, mas numa escala de 0 a 3.

Esta proposta de ação visa agilizar o processo de definição de objetivos, devido às necessidades poderem ser levantadas ao longo do trimestre, em vez de só no final. Além disso, veio dar, com antecedência, visibilidade aos PM de cada equipa, de forma a conseguirem adiantar a proposta de OKR, evitando o atraso do trimestre seguinte. Esta *framework* tem como objetivo dar resposta a C3.1 e C3.2, mitigando o atraso que acontecia todos os trimestres.

5.2 Soluções para o processo de Planeamento

Com o intuito de abordar as oportunidades de melhoria identificadas no capítulo 4.2, foram desenvolvidas as seguintes propostas de ação:

(A2) Melhoria do Issue Creator

A normalização do *Issue Creator* era uma carência existente na empresa, devido a não haver um cuidado na especificação das *issues*, e todos os tipos serem tratados do mesmo modo. A Figura 6 representa o fluxo que se seguia antes desta sugestão de melhoria.

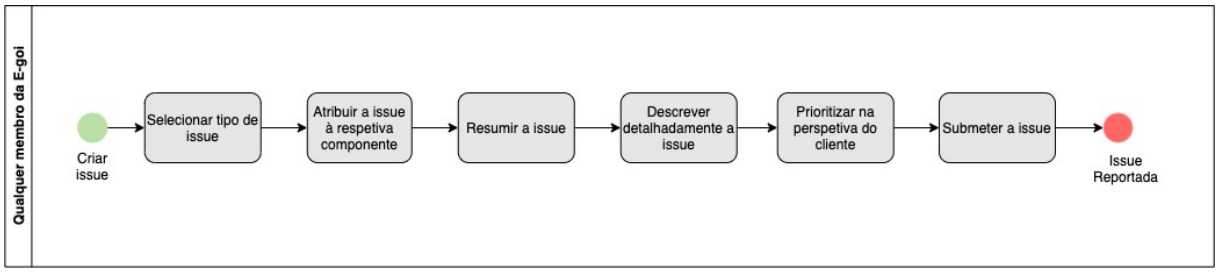


Figura 6 – Fluxo do Issue Creator As-Is

Portanto, de forma a colmatar as causas descritas no P2, está a ser desenvolvida uma melhoria do antigo *Issue Creator*, com base no fluxo representado na Figura 7, para colocar em prática a criação de *issues* mais detalhadas e evitar que a descrição seja ambígua e/ou incompleta. Esta nova ferramenta permanecerá na plataforma Admin da empresa e está dividida em três interfaces diferentes – *Bug Report, Improvement Request, Feature Request* – com os campos necessários para descrever cada tipo de *issue* (ver *Anexo I*).

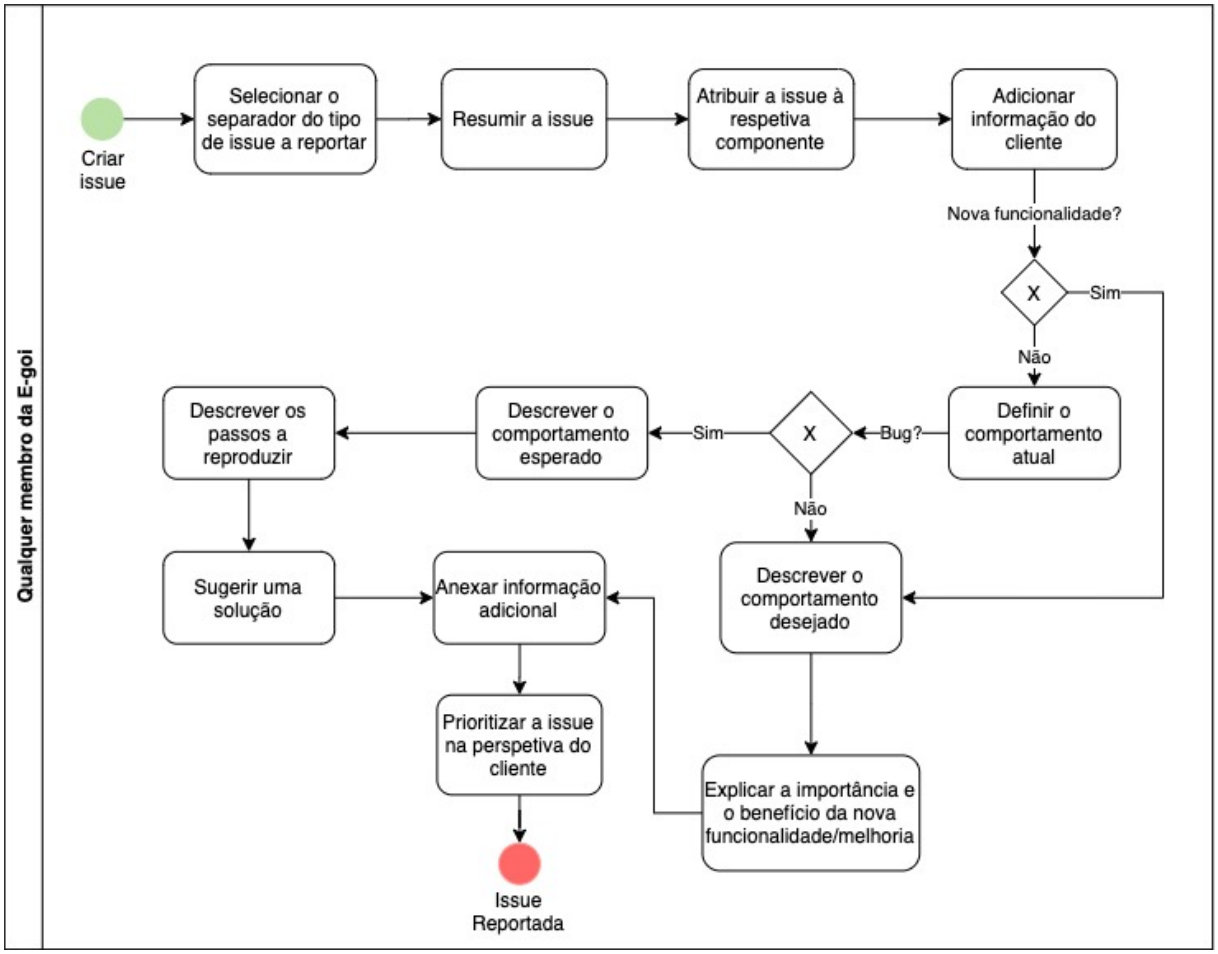


Figura 7 – Fluxo Issue Creator To-Be

No que se refere ao *bug report*, a descrição detalhada será apresentada de forma diferente, ou seja, em vez de existir um campo no qual se inseria o texto com a explicação do *bug*, foi criado um novo formato para demonstrar a informação do comportamento do respetivo. Este novo formato é constituído pelos seguintes campos:

(i) Comportamento atual: destinado para caracterizar o erro;
(ii) Comportamento esperado: serve para explicar qual deveria ser o comportamento correto da funcionalidade;
(iii) Passos para reproduzir: dá instruções passo a passo, facilitando a solução e rastreamento do erro;
(iv) *Workaround*: serve para, no caso do responsável da criação da *issue* encontrar uma solução, apresentá-la neste campo;
(v) Informação adicional: visa anexar informação adicional, incluindo capturas de ecrã, informação da base de dados, registo da consola, etc. Este campo é comum às restantes interfaces.

O *improvement request* é destinado para sugerir possíveis melhorias no produto. Tal como no *bug report*, o campo que existia para descrever detalhadamente as *issues* será substituído, de modo à recomendação ser mais compreensível e completa por parte de quem a abriu. Este novo formato inclui os seguintes campos:

(i) Comportamento atual: destinado para descrever o fluxo da funcionalidade;
(ii) Comportamento esperado: serve para explicar com detalhe a alteração pretendida;
(iii) Objetivo: tem como função descrever detalhadamente a importância e os benefícios que a melhoria poderá trazer para o cliente;

Relativamente à *feature request*, serve para os membros da empresa sugerirem novas funcionalidades. Os campos de preenchimento são os mesmos que o *improvement request*, excluindo o comportamento atual, devido a ser uma funcionalidade que ainda não existe no produto.

(A3) Sistema de identificação de issues duplicadas

Esta solução visa facilitar o processo de análise dos PM, através da implementação de um mecanismo de validação no *Issue Creator*.

Este mecanismo permite que todos os colaboradores, antes da criação de uma *issue*, explorem as já existentes, através de palavras-chave, ou seja, assim que algum membro da empresa inserir o resumo da *issue*, serão listadas todas as encontradas no JIRA que poderão estar relacionadas (ver Figura 8). A lista é constituída pelo código que está associado à *issue*, onde é possível clicar para ver a informação detalhada no JIRA, e o respetivo resumo. Além disso, no caso de serem identificadas duas *issues* repetidas, será permitido eliminar ou consolidá-las, de forma à informação reportada ser mais enriquecedora para a equipa que a vai analisar.

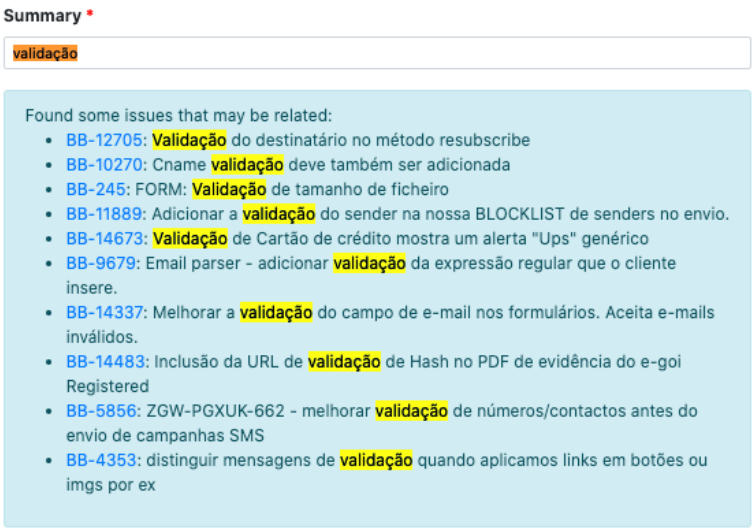


Figura 8 - Sistema de identificação de duplicação de *issues*

Esta proposta de ação contribuirá para a diminuição do número de *issues* pendentes por analisar e, consequentemente, resultará numa maior eficiência dos *Masters* a analisar as *issues*, devido a diminuir o tempo desperdiçado a examinar e/ou reproduzir os *bugs* repetidos.

(A4) Fluxo com novo estado

Para responder à C4.1, foi proposto inserir o estado "Require Prior Action" no quadro Kanban usado pelo departamento de desenvolvimento de produto. Este estado é destinado para ocasiões extremas, no qual a categorização da *issue* é a mais alta do *Backlog*, e apesar de urgente o seu desenvolvimento, este depende de um fator externo que o impede de proceder ao mesmo. As principais ocasiões, que podem levar às equipas do departamento de produto a inserir o cartão neste estado, são:

(i) A falta de recursos (Ex: servidor);
(ii) Necessidade de um melhor esclarecimento e/ou detalhe por parte de quem abriu a *issue*;
(iii) Necessidade de apoio técnico por parte do *Tech Lead* e/ou do CTO;
(iv) Dependência de outras equipas de produto e/ou core.

Portanto, este estado vai permitir aos colaboradores do departamento identificar as tarefas que requerem qualquer tipo de ação anteriormente referidas e filtrar a lista de tarefas que estão dependentes de algo externo. Este vai encontrar-se entre a coluna do *Backlog* e o *For Development*, estando representado a alteração do fluxo do quadro Kanban no *Anexo I*.

(A5) Inserção de novas *Issue Resolutions*

Como havia ocasiões em que não existiam *issue resolutions* que refletissem, para todos os membros da empresa, a análise realizada por cada *Master*, foram inseridas novas. Os *bugs* após serem analisados e/ou desenvolvidos são classificados como *Fixed* ou *Won't fix*, enquanto os restantes tipos de *issue* são especificados como *Done* e *Won't do*. Para a pessoa que criou a *issue* é essencial saber a análise realizada pela equipa atribuída. Portanto, de forma a complementar os tipos de *issues resolutions* existentes, foram acrescentadas as seguintes:

- *Fixed – Solved with reserves:* Quando o *bug* foi parcialmente resolvido para um caso específico, no entanto, pode acontecer novamente;

- *Won't do – deprecated*: A funcionalidade não vai suportar mais melhorias. Isto pode levar a outro tipo de abordagem;
- *Done – Solved in another issue*: Foi desenvolvida na resolução de outra *issue* relacionada.

No *Anexo J* encontram-se as *issues resolutions* novas, conjuntamente com as que já existiam.

5.3 Solução para o processo de Lançamento

Com o objetivo de mitigar os problemas identificados no capítulo 4.3, foi desenvolvido a seguinte proposta de ação:

(A6) O pré-lançamento fica sobre a responsabilidade do *Lead Developer* de cada equipa de produto.

Considerando o pouco conhecimento do PM perante os *commits* e épicos a serem reportados ao RM, esta responsabilidade passou a ser unicamente do *Lead Developer*. Portanto, o *Lead Developer* de cada equipa de produto deve colaborar diretamente com o RM, caso exista algo a ser lançado na nova versão. Esta solução veio minimizar o tempo que era desperdiçado, entre o PM e o *Lead Developer*, no esclarecimento de dúvidas sobre o que reportar.

5.4 Solução para problemas transversais ao Departamento de Desenvolvimento

Com o objetivo de abordar as oportunidades de melhoria identificadas no capítulo 4.4, foram sugeridas as seguintes propostas de ação:

(A7) Desenvolvimento de *workflows*

Esta proposta de ação visa normalizar as especificações fornecidas aos colaboradores do departamento de desenvolvimento de produto e entregar o serviço da empresa com qualidade.

Sendo que a falta e falha de comunicação dentro de uma empresa são os problemas que mais prejudicam o seu desempenho, tornando-o mais confuso, com o uso desta prática há um fio condutor que todos os colaboradores podem seguir, diminuindo o risco de informações descentralizadas.

Esta solução irá garantir que cada trabalhador tenha uma imagem clara do que deve ser realizado em cada fase do processo, diminuindo a existência de dúvidas e incertezas. Sendo o desenvolvimento de produto de *software* um processo colaborativo, é fundamental mostrar as interações entre os participantes, através de uma sequência de atividades, e as trocas de mensagens entres eles. Portanto, o desenvolvimento de um *workflow* agiliza a gestão de tarefas, visto que processos bem estruturados são cumpridos mais rapidamente e precisamente, oferecendo fluidez no cumprimento de tarefas. Também reduz a quantidade de erros e etapas desnecessárias porque, seguindo corretamente os passos do *workflow*, o risco de haver erros diminui (Ko, 2009).

Todos os benefícios anteriormente apresentados resultam na diminuição do volume de gastos da empresa, tanto a nível de produtividade das equipas, como consequentemente a nível económico, devido aos clientes serem impactados indiretamente com o aperfeiçoamento dos processos internos, melhorando a sua experiência.

(A8) Integrar o cliente para o desenvolvimento de novas funcionalidades

Como referido no P9, algumas equipas do departamento de desenvolvimento de produto só integram o cliente após o lançamento, não existindo contacto durante a especificação de requisitos, realizada trimestralmente, para o sistema.

De acordo com Lindberg et al. (2011), a aplicação da metodologia *Design Thinking* irá promover a aprendizagem interna, com a partilha de conhecimento entre os colaboradores e os clientes que terá impacto na experiência do cliente, contribuindo para a melhoria contínua do produto. Para articular esta metodologia com a estrutura, cultura e processos de desenvolvimento de software, é importante ter em conta os seguintes aspetos (Lindberg et al., 2011):

(i) Construir com base na diversidade: Facilitar a coesão entre as equipas e a frequente comunicação e colaboração ao longo de todo o processo de conceção;
(ii) Explorar o espaço do problema: Exploração partilhada e abrangente do problema abordado, previamente ao início do processo de desenvolvimento, através da compreensão das diferentes perspetivas do público alvo e o seu enquadramento social;
(iii) Explorar o espaço da solução: Idealização e conceptualização de diferentes ideias, de modo a descobrir uma solução sustentável para o problema;
(iv) Alinhamento de ambos os espaços: Possibilitar um processo de desenvolvimento iterativo com uma integração antecipada e contínua dos feedbacks dos utilizadores com base em protótipos compreensíveis.

Portanto, com o intuito de integrar o DT com a cultura do departamento de desenvolvimento da empresa, foi proposto acrescentar um novo processo – *Design Thinking* –, que pode ser enquadrado previamente ao processo de definição de objetivos, de forma a enquadrar o cliente desde o início do processo de desenvolvimento para auxiliar a definir futuros conceitos para o produto (cf. Figura 9). Estes conceitos podem ser novas funcionalidades ou melhoria das existentes, com o intuito de trazer valor acrescentado ao produto. Este tipo de abordagem é uma mais valia, visto ser focada no utilizador, sendo este o principal ativo da empresa.

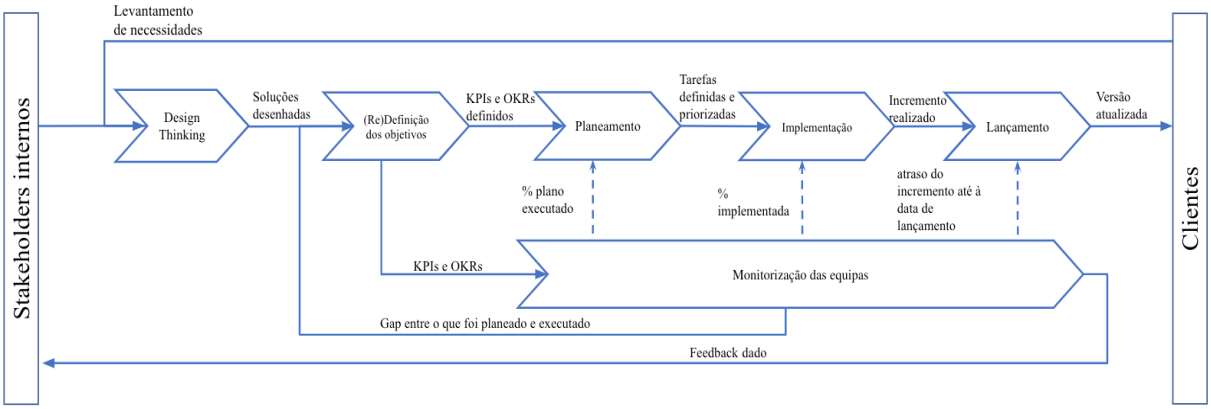


Figura 9 - Mapa de Processos *To-Be*

De forma a compreender as alterações fruto da inserção do DT, foi desenhada a matriz RACI para demonstrar as fases deste processo e o seu respetivo resultado (Cf. Figura 10). Este modelo não representa o funcionamento na sua totalidade devido aos processos não serem desempenhados de uma forma linear.

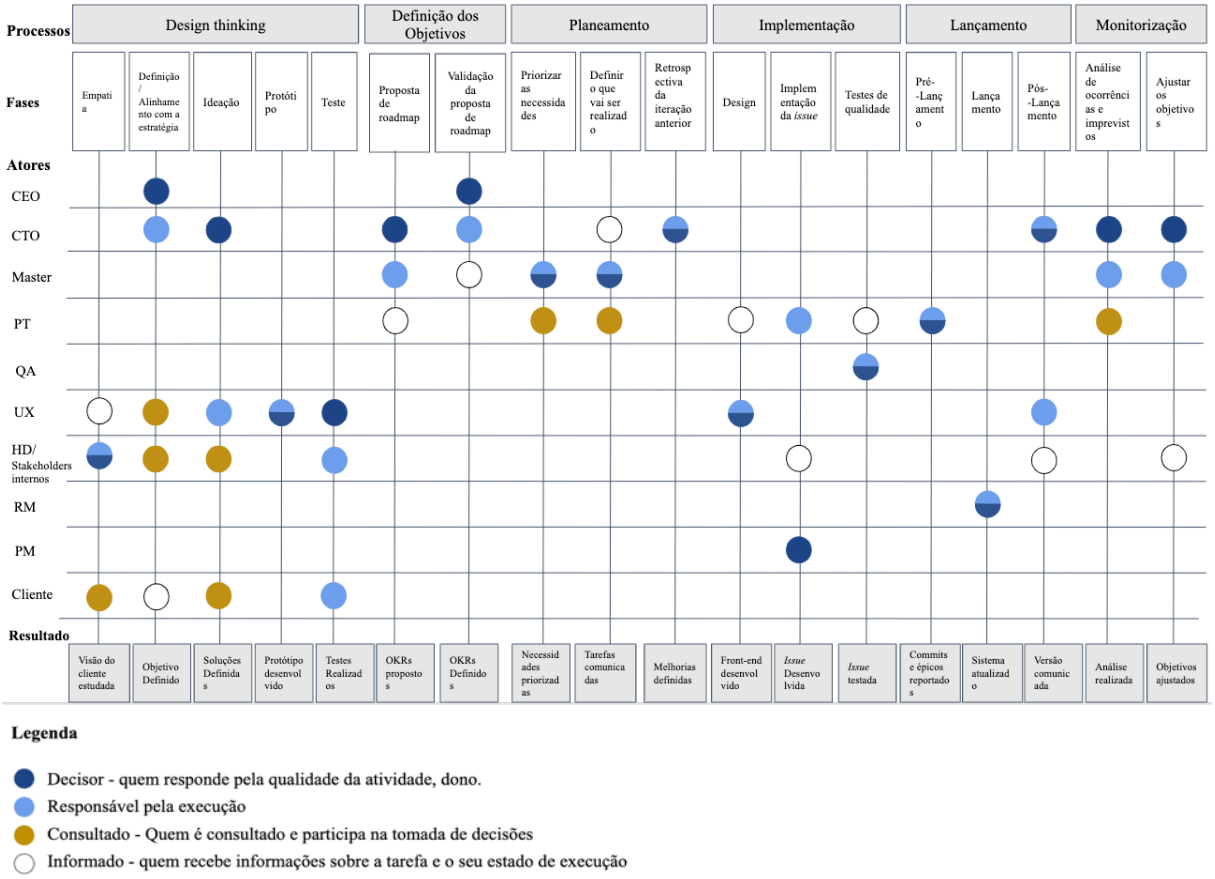


Figura 10 - Matriz RACI *To-Be*

A primeira fase deste processo será levantar informação disponível, relativa às necessidades dos utilizadores e o contexto no qual estão inseridos. Isto implica envolver especialistas para compreender o contexto em que as pessoas envolvidas se encontram, as suas experiências e motivações nas atividades, e as tecnologias necessárias para a realização das mesmas. Esta fase será liderada pela equipa de Suporte de Apoio ao Cliente *Outbound*, na medida em que estes, no dia-a-dia, têm contacto direto com o cliente, no suporte das suas necessidades perante o produto.

Com a informação recolhida referente às características dos utilizadores, segue-se a fase de definição, que consistirá na sistematização do problema e no delineamento mais específico da situação que se pretende resolver através de entrevistas realizadas aos utilizadores. Nesta fase seria importante a colaboração do CEO, para clarificar quais as necessidades mais prioritárias, alinhando-as com a estratégia da empresa.

Subsequentemente à definição do problema concluída, a fase de ideação é dedicada ao debate de múltiplas ideias para a solução dos respetivos problemas encontrados na fase anterior. Nesta etapa é aplicado o pensamento divergente para adquirir o maior número de ideias possíveis, para que, de seguida, através do pensamento convergente, selecionar as mais viáveis considerando o impacto que as respetivas terão no sistema. Assim, é importante que nesta fase

esteja presente o CTO, devido à sua visão geral sobre o sistema, validando o prosseguimento das diferentes soluções para a realização do protótipo.

A fase de prototipagem é fundamental para testar as soluções selecionadas com os diferentes *stakeholders* e/ou clientes, cujo objetivo é avaliá-las conforme a experiência dos utilizadores. No fim desta etapa, será mais percetível as restrições inerentes ao produto, os problemas ainda existentes e a informação relativa ao comportamento dos utilizadores durante a sua interação.

A última fase deste processo é dedicada a testar as soluções desenvolvidas durante a fase de prototipagem. Dependendo do resultado dos testes, sendo o DT um processo iterativo, é frequente o surgimento de mais problemas, pelo que é natural serem reveladas novas ideias por parte dos utilizadores, o que leva a retornar novamente à fase de ideação ou à de prototipagem.

Na Tabela 7, estão enumeradas técnicas comuns que podem ser enquadradas ao longo das fases desta metodologia.

Tabela 7 - Técnicas para cada fase do processo de Design Thinking (Tschimmel, 2012)

Fase	**Técnicas**	**Descrição**
Empatia e Definição do problema	Observação	Fornece informações para a definição dos perfis do utilizador. Estas informações podem ser adquiridas através de fotografias, gravação do comportamento padrão das pessoas.
	Mind Maps	É o mapeamento e organização sistemática de informação, de uma forma visual, da informação recolhida na observação e nas entrevistas aos utilizadores, com o intuito de explorar padrões e extrair significados (Ex: Diagramas de afinidade, mapas de empatia, mapa de viagem, etc.).
	Personas / Mapa de empatia	As personas são personagens fictícias que contêm atributos do utilizador de um serviço ou produto e permitem a criação de ideias, para melhorar os vários tipos de experiências. O Mapa de Empatia é uma ferramenta para organizar a informação adquirida a partir de persona e/ou das entrevistas e da observação dos utilizadores, cujo objetivo é refletir e discutir as suas necessidades, emoções, desejos e medos, relacionados com o âmbito do projeto.
Ideação	Brainwriting	Produção de ideias em post-its por parte de todos os participantes do processo através da escrita ou de desenhos.
	Esboços	Gerar novas ideias e perspetivas a partir de desenhos simples para explicar, clarificar e discutir ideias, sendo essencial para torná-las tangíveis e concretas.
	Confronto Visual e Semântico	Associação entre um problema, ou tarefa, e palavras, frases, imagens, fotografias, etc. Quanto mais distante for a sua relação,

		melhor será o resultado. (Ex. Intuição Semântica)
Protótipo	*Wireframes*	Desenvolvimento rápido de algo material que possa facilitar a discussão entre os *stakeholders* e posteriormente testado com os utilizadores. Após os testes, o protótipo pode ser aperfeiçoado.
	Storyboard	Baseia-se na criação de uma história através de uma sequência de imagens para exibir um processo, serviço ou evento, com o objetivo de testar a solução.

A integração *Design Thinking* terá impacto na definição dos objetivos, uma vez que os *stakeholders* internos terão um maior envolvimento para colaborarem tanto durante a fase de definição do problema como na sugestão de ideias, havendo benefícios, incluindo o conhecimento destes em variadas áreas e pela visão que estes têm perante a plataforma pelo facto de a utilizarem diariamente. Além disso, como o cliente é o elemento chave neste processo, poderá relatar os possíveis problemas identificados e sugerir novas ideias que melhorem a performance e usabilidade do produto. Portanto, a exploração, recolha e análise de dados que é realizada ao longo do processo DT conjuntamente com os stakeholders internos e clientes poderá melhorar a especificação dos requisitos para o sistema que está relacionado com este processo.

Em síntese, a Tabela 8 estabelece a relação entre as ações sugeridas e os problemas identificados no capítulo 4, demonstrando aspetos dos processos de definição de objetivos, planeamento e lançamento que podem ser melhorados e implementados no contexto da empresa.

Tabela 8 - Correspondência entre os problemas e as ações

Processo	Problema	Causa	Ação
Definição dos Objetivos	P1	C1.1	A1, A8
		C1.2	
Planeamento	P2	C2.1	A2
		C2.2	
	P3	C3.1	A3
		C3.2	
	P4	C4.1	A4
		C4.2	A5
Lançamento	P5	-	A6
	P6	C6.1	-
		C6.2	-
	P7	C7.1	-
Considerações Adicionais	P8	C8.1	A7
	P9	C9.1	A8
		C9.2	

6 Conclusões e perspetivas de trabalho futuro

O principal objetivo desta dissertação englobou o estudo do funcionamento atual da empresa, identificação das oportunidades de melhoria, desenvolvimento e implementação de soluções inerentes à Engenharia de Serviços e Gestão, de forma a colmatar os problemas identificados e serem posteriormente adaptadas à realidade empresarial.

Este projeto teve como fundamento a análise e melhoria dos processos transversais às equipas de desenvolvimento da empresa, desde o levantamento de necessidades até ao lançamento de uma nova versão do produto.

Tendo em conta que as atividades realizadas durante todos os processos do departamento de desenvolvimento de produto não tinham qualquer tipo de procedimento visual, dependendo apenas do conhecimento e experiência dos colaboradores atuais, a realização de entrevistas informais, observação direta e análise documental existente serviu como ponto de partida, proporcionando um conhecimento aprofundado sobre o funcionamento do departamento e a forma como os processos estavam organizados. Com base na informação recolhida, foram identificados cinco processos críticos – Definição dos objetivos, Planeamento, Implementação, Lançamento e Monitorização das equipas -, no qual o nível de atuação circunscreveu os processos de definição de objetivos, planeamento e lançamento.

A utilização de técnicas de modelação de processos foi imprescindível na medida em que permitiu ter uma base estruturada de raciocínio e, complementarmente, contribuir para a obtenção de uma visão global, que facilitou o entendimento da conexão entre as diferentes fases e o fluxo compreendido entre elas. Para possibilitar esta visão sobre o departamento de desenvolvimento de produto foi utilizado o mapa de processos que, simultaneamente, auxiliou na delimitação do nível de intervenção; o desenho multinível foi bastante útil para uma análise aprofundada do mapa de processos, através da matriz RACI, que proporcionou a definição do nível de intervenção das partes envolvidas em cada fase do processo e o respetivo resultado, e dos diagramas *swimlane,* que em conjunto com os respetivos fluxogramas foram fundamentais para compreender o fluxo de trabalho e a interação existente entre os diversos participantes de cada processo.

Seguidamente ao desenho dos processos procedeu-se à análise dos mesmos, considerando o estado de arte e a recolha de informação obtida nas entrevistas e nas conversas informais com os colaboradores da empresa. Esta informação permitiu identificar oportunidades de melhoria, nomeadamente os problemas existentes e as respetivas causas, respondendo assim à QP1. Os problemas detetados derivaram da falta de ferramentas sistematizadas, uniformização de processos e a escassa colaboração entre o cliente e as equipas de produto durante o processo, que se verificaram prejudiciais no desempenho do departamento.

Após a identificação e caracterização das oportunidades de melhoria, a etapa subsequente consistiu no desenvolvimento de possíveis ações que mitigassem os problemas encontrados, por intermédio de realização de workshops e sessões de brainstorming com as partes envolvidas, respondendo à QP2.

Para melhorar o processo de definição dos objetivos e com o intuito de reduzir o atraso dado no início de cada trimestre, procedeu-se ao desenvolvimento da matriz de origem-destino. Com a sua implementação espera-se dinamizar o trabalho do CTO, na reunião com os HD, para torná-la mais objetiva, reservar mais tempo para a discussão do objetivo das necessidades já levantadas e definição do grau de prioridade de cada equipa. Além disso, com esta *framework*

pretendeu-se agilizar as equipas do departamento de desenvolvimento de produto para propor os OKR's.

O impacto das melhorias executadas no âmbito do processo de planeamento não foi completamente determinado, dado o curto período de tempo. No entanto, o *feedback* obtido foi positivo, revelando que geraram uma melhor organização, nomeadamente a nível da análise das *issues*, fruto da diminuição do número de intervenções não rentáveis e de perdas de tempo que aconteciam anteriormente, proporcionando uma maior agilidade aos *Masters*. Com a sistematização do *Issue Creator* espera-se uma melhor gestão, particularmente no controlo do *Backlog*, uma vez que o número de *issues* por analisar tende a diminuir.

De modo a melhorar as taxas de conversão e retenção do cliente, foi sugerida a adição do *Design Thinking* no processo de desenvolvimento, que inclui o cliente desde o início do processo, a fim de produzir um resultado mutuamente valorizado. Como esta não foi implementada, foram sugeridas, com base nas funções exercidas de cada interveniente, as responsabilidades por cada fase e possíveis técnicas a serem usufruídas ao longo do processo, visto que é um processo demorado e necessita de colaboradores capacitados para o seu bom funcionamento.

A participação dos colaboradores da empresa foi crucial para a investigação qualitativa, visto que a normalização do processo e as melhorias desenvolvidas tiveram em consideração as pessoas e a sua concordância com o que foi documentado.

Em suma, o contributo final desta dissertação foi ajudar a operacionalizar processos do departamento de desenvolvimento que são transversais a todas as equipas e a agilizar a sua gestão, considerando que faltava dinâmica na mesma, fruto da falta de maturidade nos processos, à carência de ferramentas de apoio necessárias ou falta de sistematização das existentes.

6.1 Limitações e perspetivas de trabalho futuro

Ao longo do desenvolver do trabalho, surgiram limitações, nomeadamente a não implementação de algumas propostas de melhoria e a falta de obtenção de resultados das que foram implementadas. No entanto, a investigação realizada ao longo deste trabalho pode servir como base para futuros projetos internos, destinados a aumentar ainda mais a eficiência e a produtividade das equipas do departamento de desenvolvimento de produto. A não obtenção de resultados sobre as medidas implementadas deveu-se essencialmente ao facto de estas soluções terem impacto a longo prazo, sendo necessário a espera para os resultados serem visíveis.

A normalização que foi realizada nos processos de definição de objetivos, planeamento e lançamento ainda carece de volume e maturidade para ser percetível se, com o tempo, continua a responder ou a mitigar os problemas anteriormente identificados. Sugere-se que em futuras investigações se explorem as ferramentas desenvolvidas para a aplicação na gestão de equipas de software, com o intuito de aumentar a sua dinâmica.

Além disso, outro tópico importante seria realizar um estudo aprofundado da adoção do *Design Thinking* nos processos inerentes ao departamento de desenvolvimento de produto, visto que este não foi implementado. Esta análise pode ser realizada com base no estado de arte desta metodologia, comparando casos bem sucedidos e, por outro lado, os sem êxito, de modo a que a empresa possa tirar o máximo proveito da mesma.

De forma global, pode-se salientar que é essencial para qualquer empresa a constante análise dos seus processos e melhoria dos mesmos, de forma a aumentar a capacidade de resposta perante o mercado que se encontra. Este é um projeto que necessita de um estudo contínuo, para que se mantenha atualizado e que acrescente valor à empresa.

Por fim, como esta dissertação foi conduzida pelo método de pesquisa qualitativa, a quantificação das soluções desenvolvidas foi inexistente. Neste sentido, as futuras direções incluem uma investigação quantitativa, com o intuito de complementar a informação e as soluções sugeridas.

Referências

Ahmad, M. O., Markkula, J., & Oivo, M. (2013). *Kanban in software development: A systematic literature review.* Paper presented at the 2013 39th Euromicro conference on software engineering and advanced applications.

Aitken, A., & Ilango, V. (2013). *A comparative analysis of traditional software engineering and agile software development.* Paper presented at the 2013 46th Hawaii International Conference on System Sciences.

Awad, M. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*, 30.

Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management, 2*(1), 26-30.

Beckman, S. L., & Whalen, E. (2018). *Innovation, co-creation, and design thinking: How salesforce's ignite team accelerates enterprise digital transformation*: The Berkeley-Haas Case Series. University of California, Berkeley. Haas ….

Carbonell, P., Rodríguez-Escudero, A. I., & Pujari, D. (2009). Customer involvement in new service development: An examination of antecedents and outcomes. *Journal of product innovation management, 26*(5), 536-550.

Chinosi, M., & Trombetta, A. (2012). BPMN: An introduction to the standard. *Computer Standards & Interfaces, 34*(1), 124-134.

Cho, J. (2008). Issues and Challenges of agile software development with SCRUM. *Issues in Information Systems, 9*(2), 188-195.

Choy, L. T. (2014). The strengths and weaknesses of research methodology: Comparison and complimentary between qualitative and quantitative approaches. *IOSR Journal of Humanities and Social Science, 19*(4), 99-104.

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2013). *Fundamentals of business process management* (Vol. 1): Springer.

Gurusamy, K., Srinivasaraghavan, N., & Adikari, S. (2016). *An integrated framework for design thinking and agile methods for digital transformation.* Paper presented at the International Conference of Design, User Experience, and Usability.

Hibbs, C., Jewett, S., & Sullivan, M. (2009). *The art of lean software development: a practical and incremental approach*: " O'Reilly Media, Inc.".

Hoda, R., Noble, J., & Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and software technology, 53*(5), 521-534.

Ko, R. K. (2009). A computer scientist's introductory guide to business process management (BPM). *XRDS: Crossroads, The ACM Magazine for Students, 15*(4), 11-18.

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing, 43*, 59-67.

Lindberg, T., Meinel, C., & Wagner, R. (2011). Design thinking: A fruitful concept for it development? In *Design thinking* (pp. 3-18): Springer.

MacCormack, A. (2001). How internet companies build software. *MIT Sloan Management Review, 42*(2), 75-84.

Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015). Empirical study of agile software development methodologies: A comparative analysis. *ACM SIGSOFT Software Engineering Notes, 40*(1), 1-6.

Middleton, P. (2001). Lean software development: two case studies. *Software Quality Journal, 9*(4), 241-252.

Moniruzzaman, A., & Hossain, D. S. A. (2013). Comparative study on agile software development methodologies. *arXiv preprint arXiv:1307.3356*.

Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science Issues (IJCSI), 7*(5), 94.

Nerur, S., & Balijepally, V. (2007). Theoretical reflections on agile development methodologies. *Communications of the ACM, 50*(3), 79-83.

Nurmuliani, N., Zowghi, D., & Williams, S. (2006). *Requirements volatility & its impact on change effort: Evidence based research n software development projects.* Paper presented at the Verified OK.

Ohno, T. (1988). *Toyota production system: beyond large-scale production*: crc Press.

Petersen, K. (2010). *Implementing lean and agile software development in industry.* (Doctoral Thesis). Blekinge Institute of Technology,

Poppendieck, M., & Cusumano, M. A. (2012). Lean software development: A tutorial. *IEEE software, 29*(5), 26-32.

Rahim, M. S., Chowdhury, A. E., & Das, S. (2017). *Rize: A proposed requirements prioritization technique for agile development.* Paper presented at the 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC).

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal, 20*(5), 449-480.

Royce, W. W. (1987). *Managing the development of large software systems: concepts and techniques.* Paper presented at the Proceedings of the 9th international conference on Software Engineering.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*: Addison-Wesley.

Scrum.org. (2018). What is Scrum? Retrieved from https://www.scrum.org/resources/what-is-scrum

Sokovic, M., Pavletic, D., & Pipan, K. K. (2010). Quality improvement methodologies–PDCA cycle, RADAR matrix, DMAIC and DFSS. *Journal of achievements in materials and manufacturing engineering, 43*(1), 476-483.

Taylor, G. R. (2005). *Integrating quantitative and qualitative methods in research*: University press of America.

Tschimmel, K. (2012). *Design Thinking as an effective Toolkit for Innovation.* Paper presented at the ISPIM Conference Proceedings.

VersionOne. (2019). 13th Annual State of Agile Report. Retrieved from https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508

Wang, X., Conboy, K., & Cawley, O. (2012). "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of systems and software, 85*(6), 1287-1299.

White, S. A. (2004). Introduction to BPMN. *Ibm Cooperation, 2*(0), 0.

Womack, J. P., & Jones, D. T. (1996). Beyond Toyota: how to root out waste and pursue perfection. *Harvard business review, 74*(5), 140-&.

ANEXO A: Guiões de Entrevistas

Guião para os *Product Managers*

1. Qual é a função de um *product manager*?
2. Quais são as tuas maiores dificuldades na gestão de equipa?
3. Como é que os objetivos são definidos ao longo do tempo?
4. Como é que é realizado o planeamento?
5. Como é que são organizadas a lista de *issues*? Quando e como é que são comunicadas à equipa de desenvolvimento?
6. Como é realizada a priorização das tarefas/funcionalidades?
7. Quando existem interdependência de tarefas, qual a forma de as otimizar?
8. Que medidas de performance são utilizadas para medir os indicadores de desempenho processo de desenvolvimento?
9. Como e de que forma é realizado o lançamento?
10. Após o lançamento de uma nova funcionalidade como é que é transmitido ao cliente?

Guião para as Equipas Core

Questões comuns a todas as equipas core

1. Qual é o teu papel na empresa?
2. Quais são as tuas responsabilidades diárias?

Quality Assurance

1. Conta-me como é que identificaste o último problema/tarefa que te apareceu. Quais são as fases que constituem este processo?
2. Quais são o tipo de testes realizados para garantir uma boa qualidade de software?
3. Quando é que é realizado o controlo de qualidade? Imediatamente após o desenvolvimento de uma funcionalidade/*issue*?
4. Como é que priorizam as *issues* que têm que testar?
5. Quanto tempo em média demora a resolver (problema/tarefa)?

User Experience

1. Quais são as técnicas usadas para realizar o *User Research*?
2. Quais são os métodos utilizados para obterem o máximo de feedback?
3. Como recebem as *strings*? Qual tem maior prioridade? Quanto tempo em média demoram a executá-las?

DevOps & Architecture

1. Como é feita a análise do código-fonte?
2. Considerando que és responsável pelo processo de lançamento, pretendia perceber o fluxo deste processo.
3. Em que ocasiões as equipas de produto te costumam consultar e vice-versa?
4. Como é que as diferentes partes do produto se relacionam?
5. Na tua perspetiva, o que consideras que pode melhorar no ciclo de desenvolvimento de produto de software?

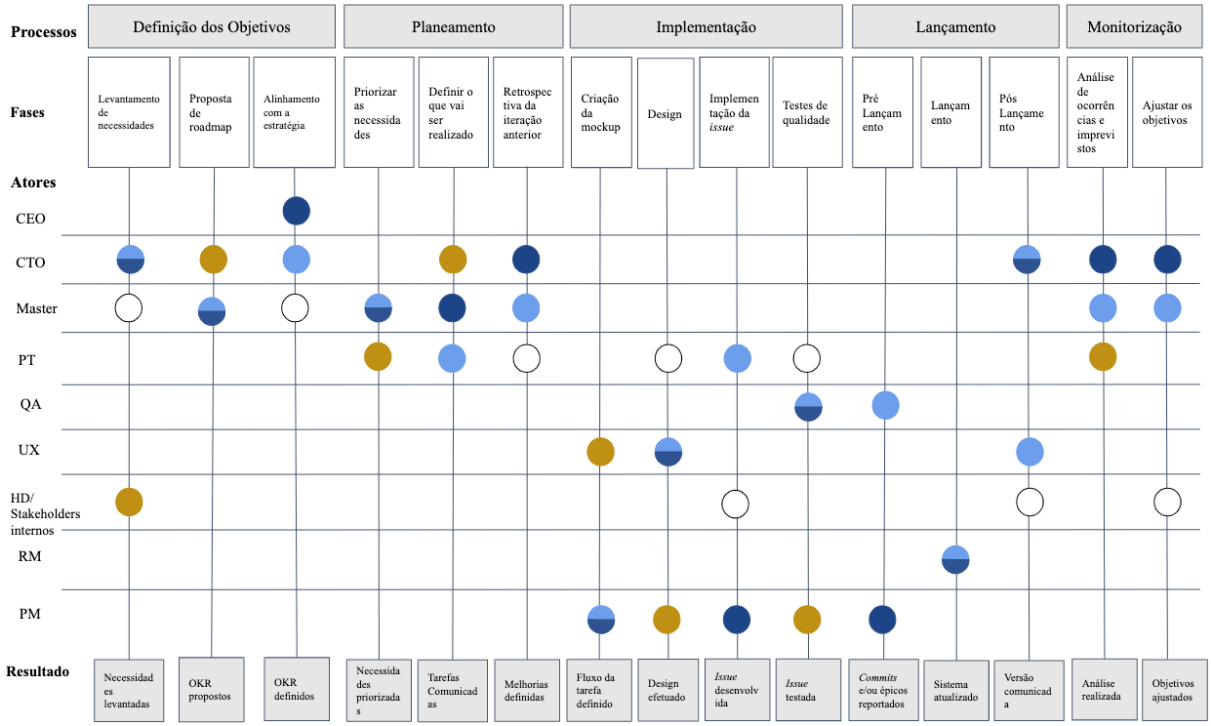
SysOps & Architecture

1. Quando e como deve ser feita a migração de uma aplicação para os servidores da E-goi?
2. Em que circunstâncias o Dep. de Desenvolvimento de produto costuma abordar-vos? Quais os motivos?
3. Que métricas são usadas para monitorar a performance e disponibilidade do sistema?
4. Quais são os processos de entregabilidade existentes?

Guião para os *stakeholders* internos

1. Com que frequência costumam comunicar com o departamento de desenvolvimento de produto? Qual costuma ser o motivo?
2. Quais são as formas de comunicação existentes entre o vosso departamento e o de produto? São eficientes?
3. Quais são as fraquezas do departamento de desenvolvimento de produto? E o que pode ser melhorado?
4. Qual o impacto que uma atualização do produto pode ter no seu departamento?
5. Qual a dinâmica existente para responder às necessidades dos clientes?
6. Quais são as maiores dificuldades em relação ao departamento de desenvolvimento de produto?

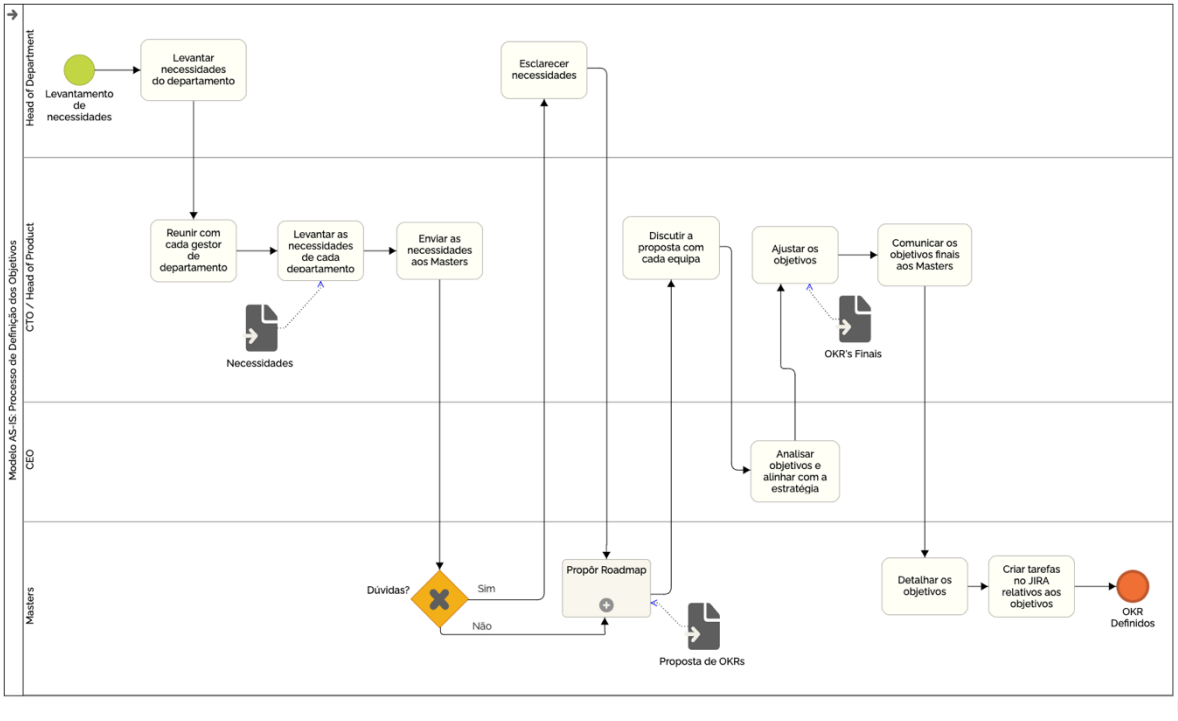
ANEXO B: Matriz RACI *As-Is*

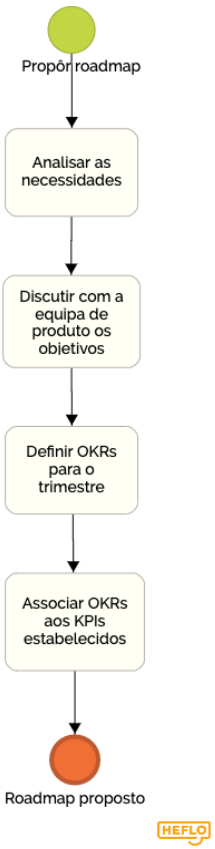


Legenda

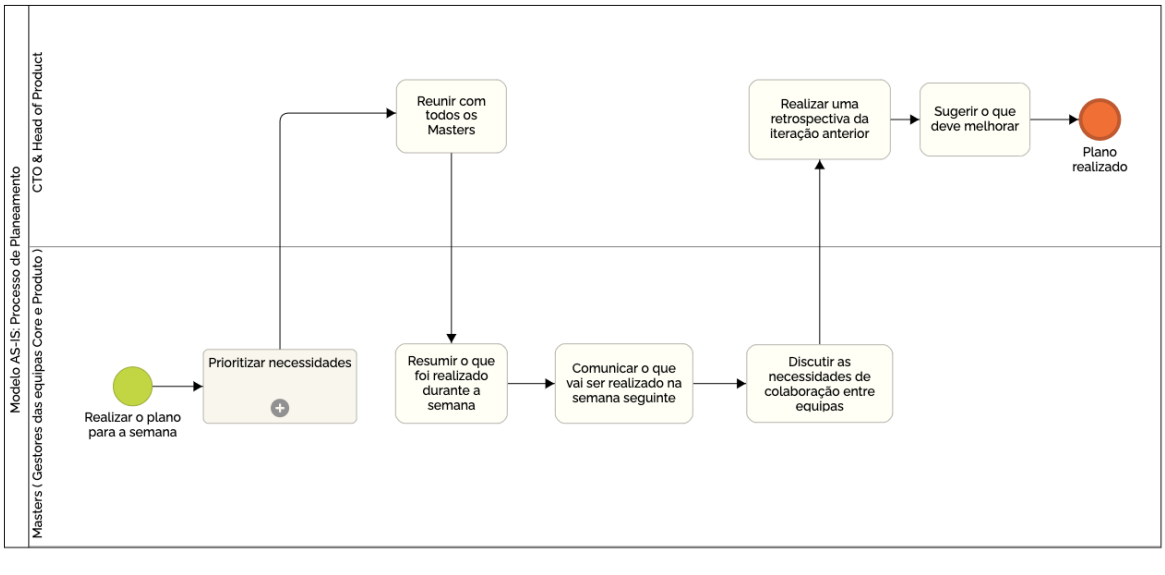
- Decisor - quem responde pela qualidade da atividade, dono.
- Responsável pela execução
- Consultado - Quem é consultado e participa na tomada de decisões
- Informado - quem recebe informações sobre a tarefa e o seu estado de execução

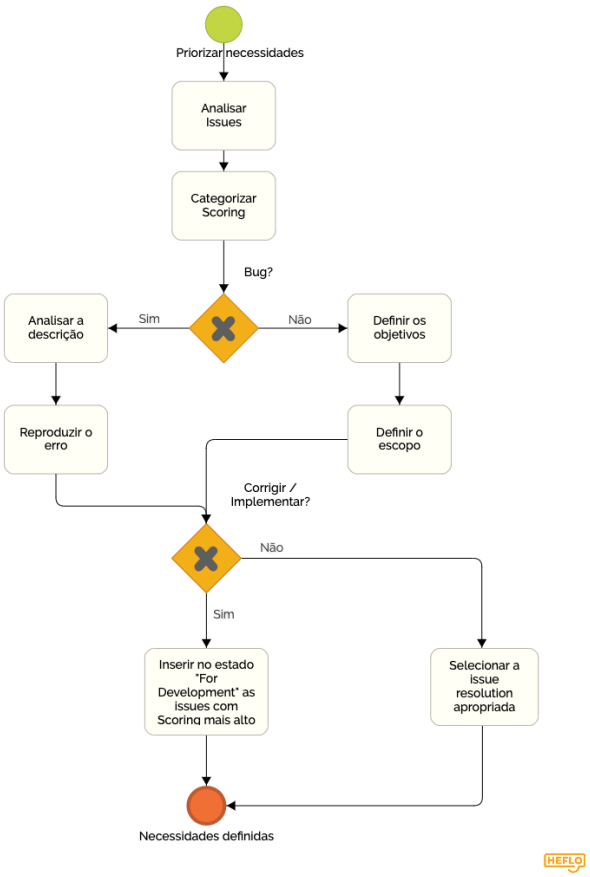
ANEXO C: Processo de Definição dos Objetivos



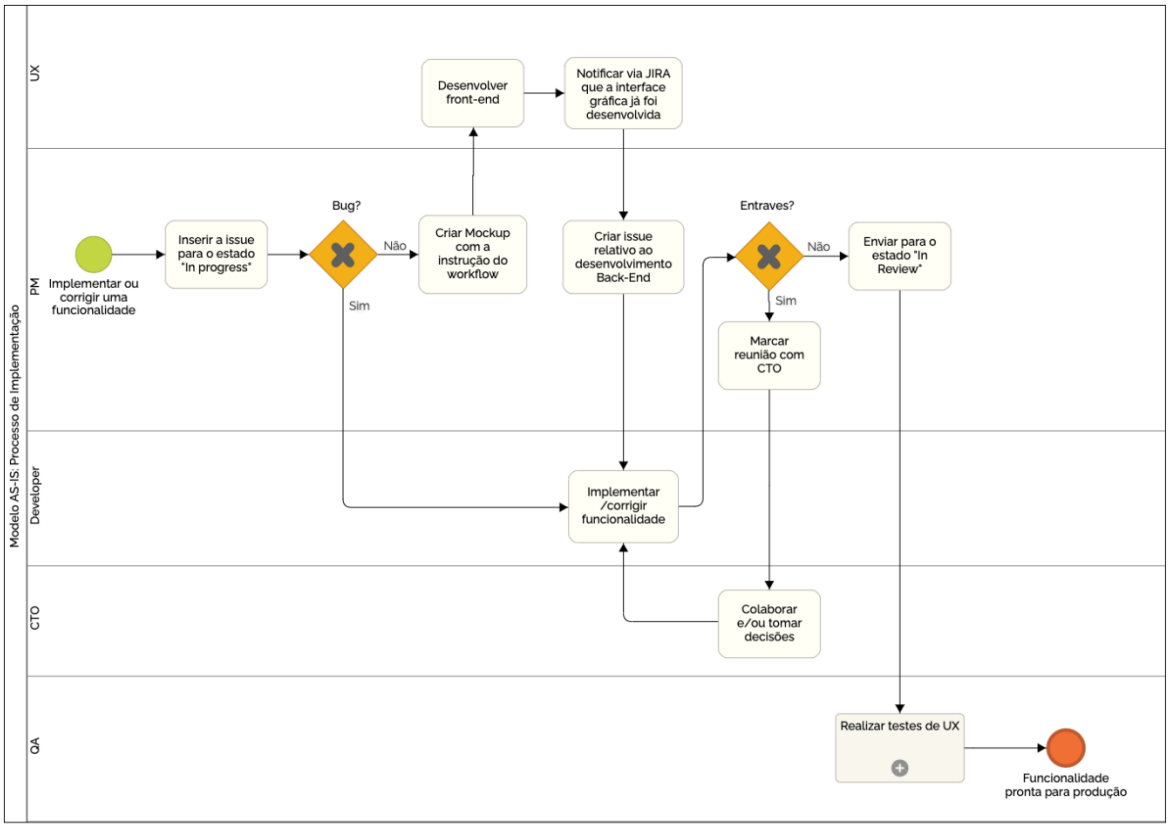


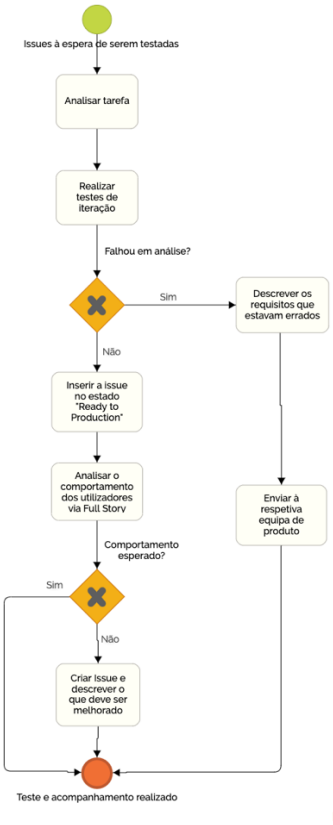
ANEXO D: Fluxo do Processo de Planeamento



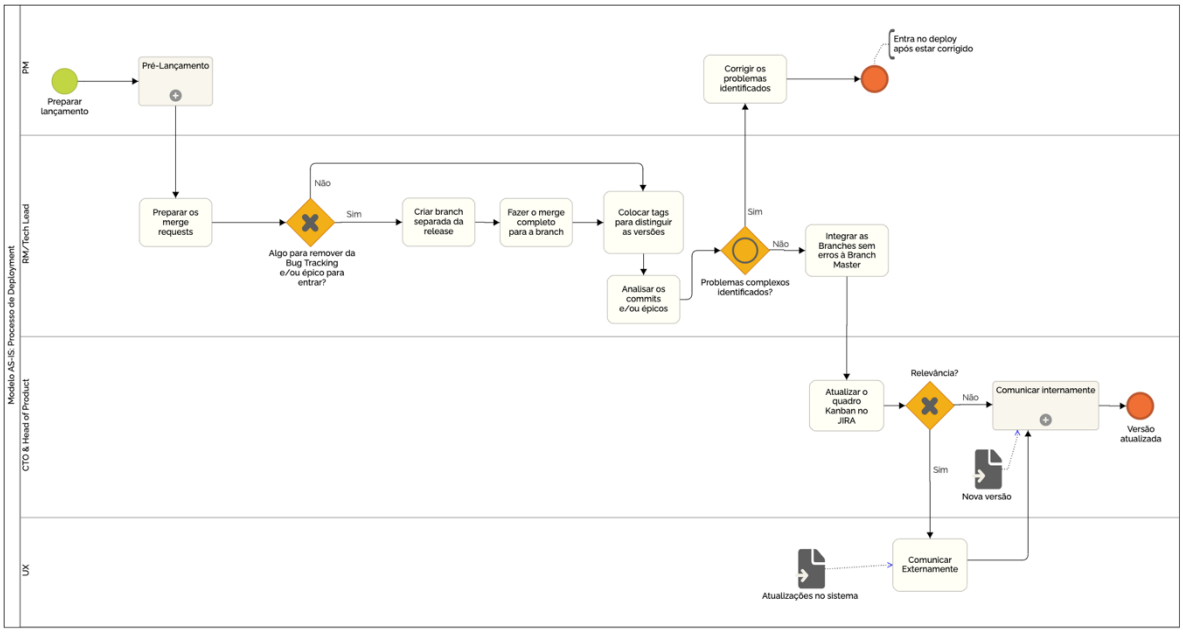


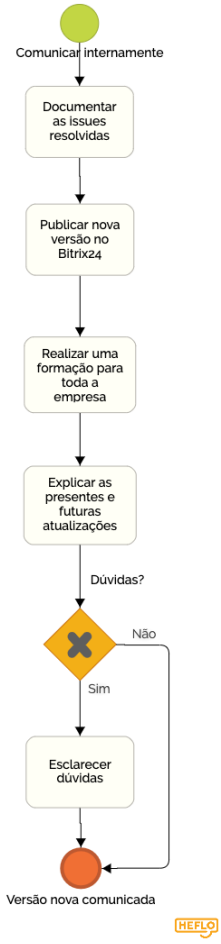
ANEXO E: Fluxo do Processo de Implementação



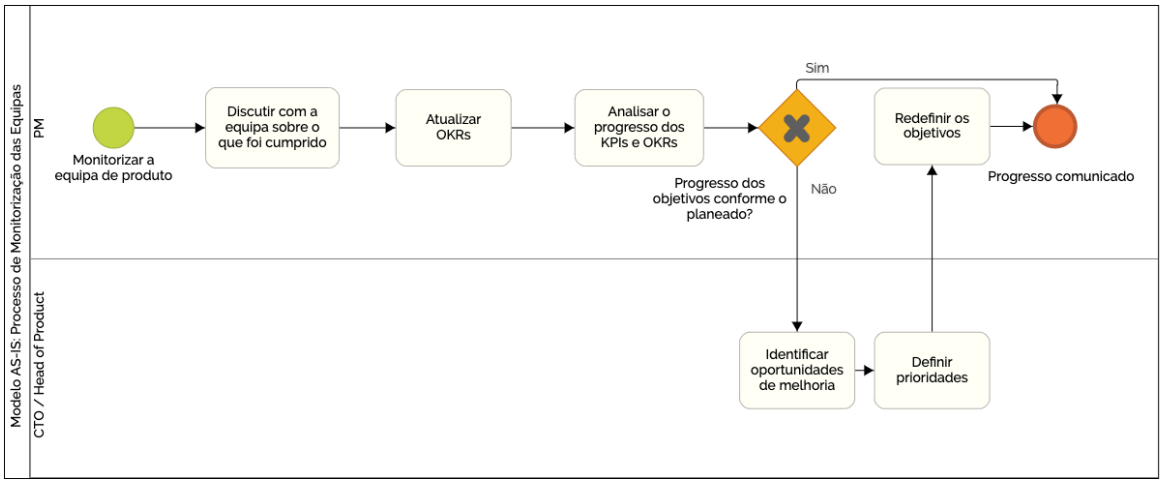


ANEXO F: Fluxo do Processo de Lançamento

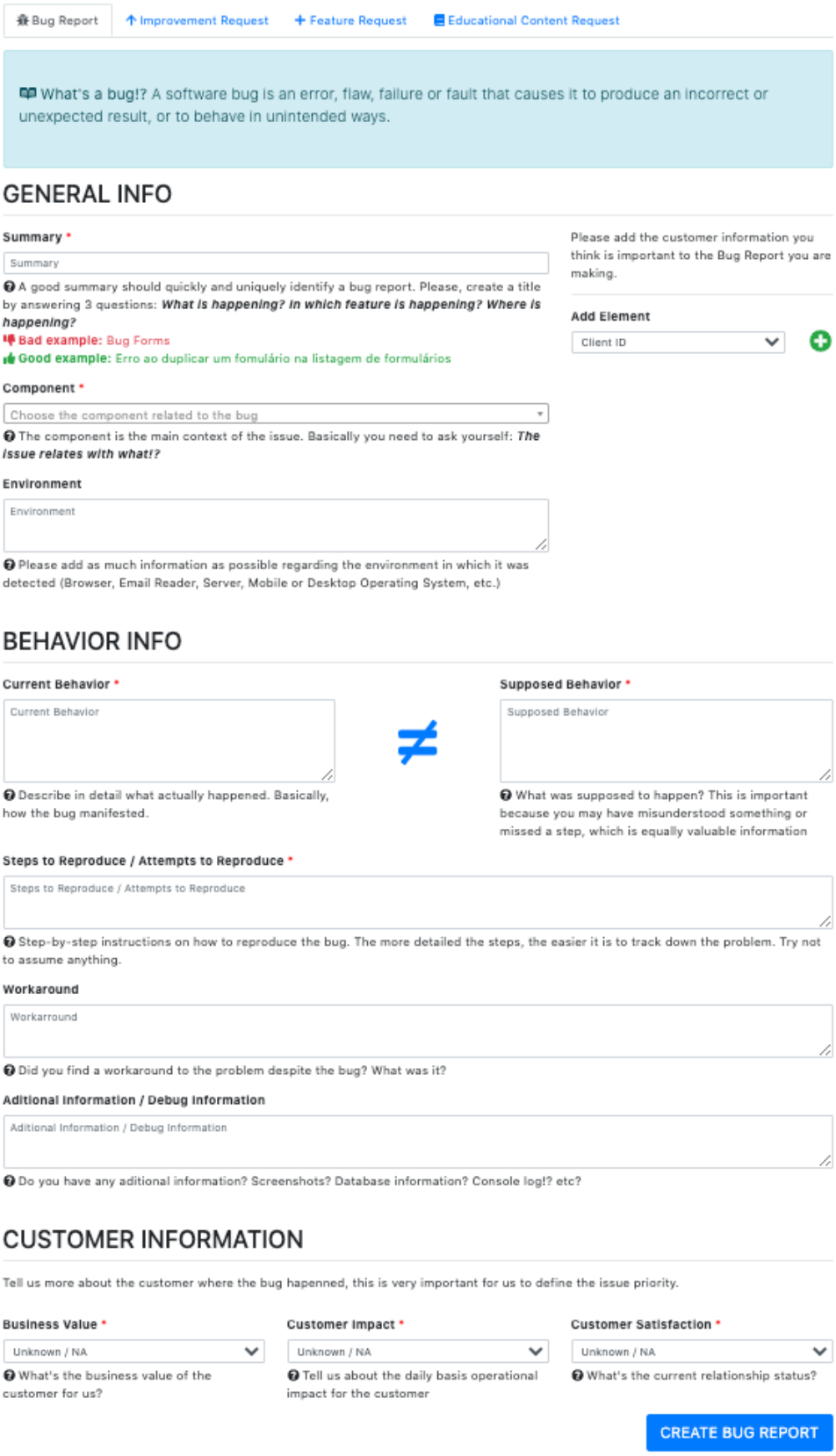


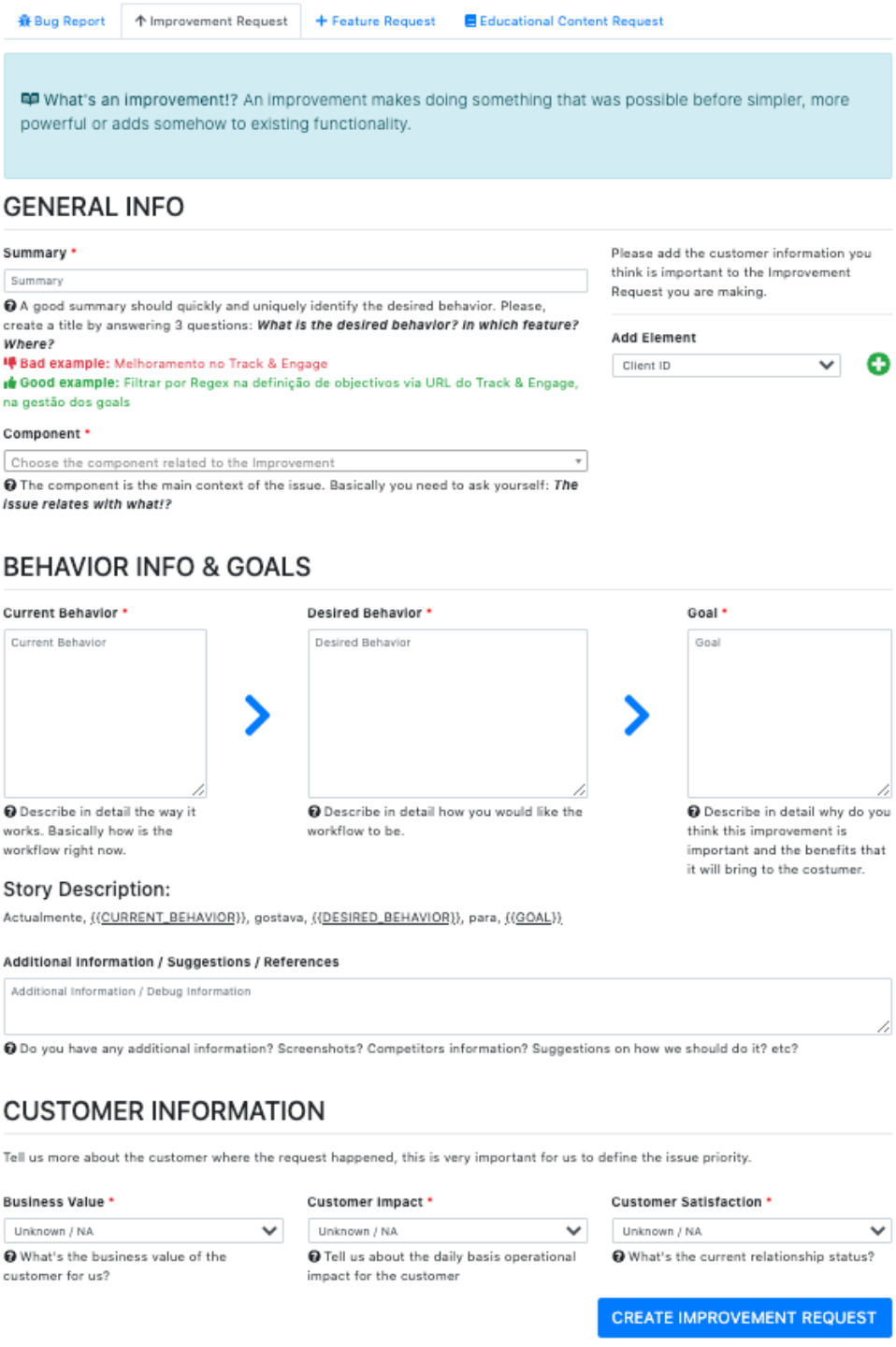

Comunicar internamente
Documentar as issues resolvidas
Publicar nova versão no Bitrix24
Realizar uma formação para toda a empresa
Explicar as presentes e futuras atualizações
Dúvidas?
Não
Sim
Esclarecer dúvidas
Versão nova comunicada
HEFLO

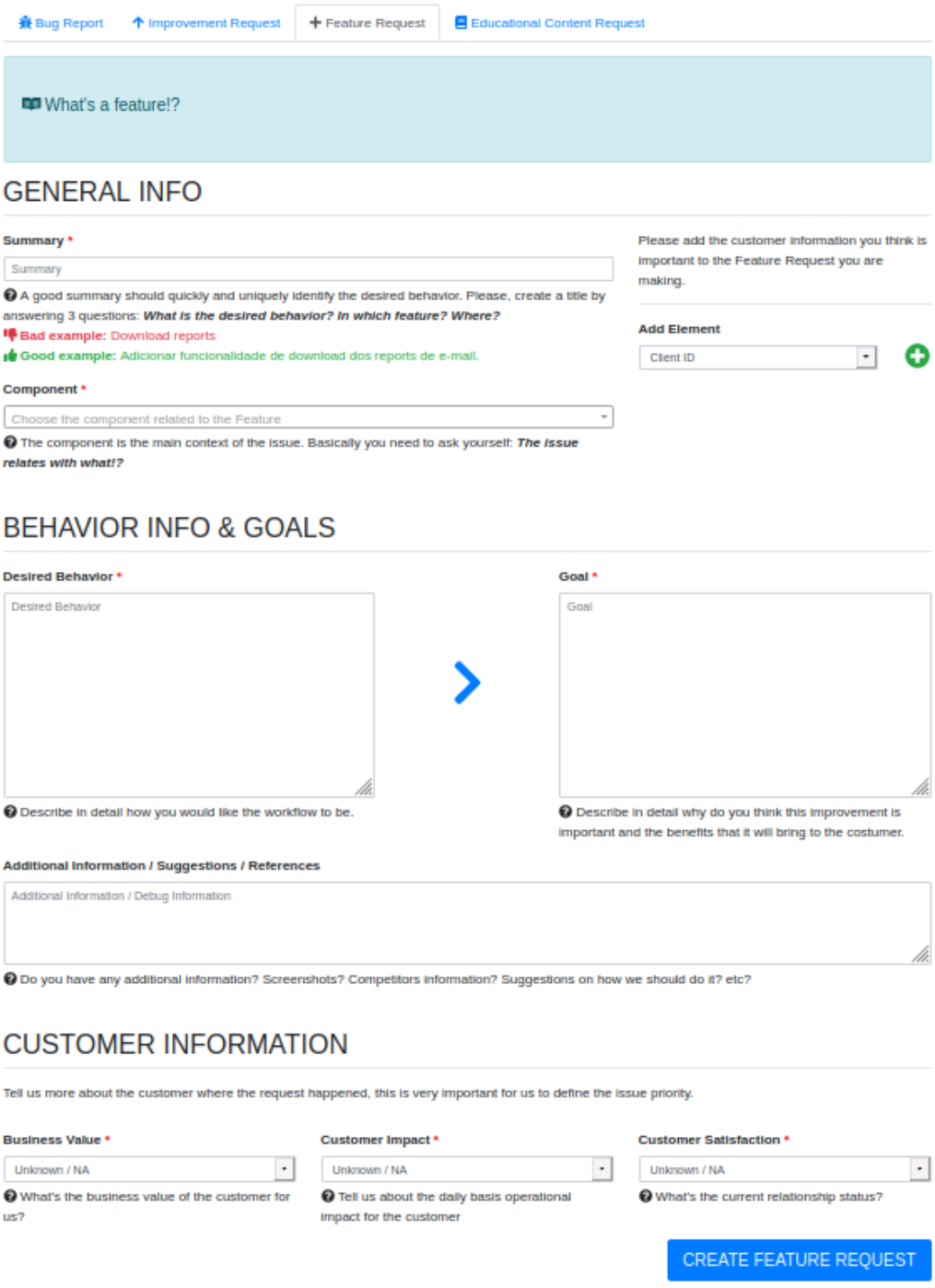
ANEXO G: Fluxo do Processo de Monitorização das Equipas



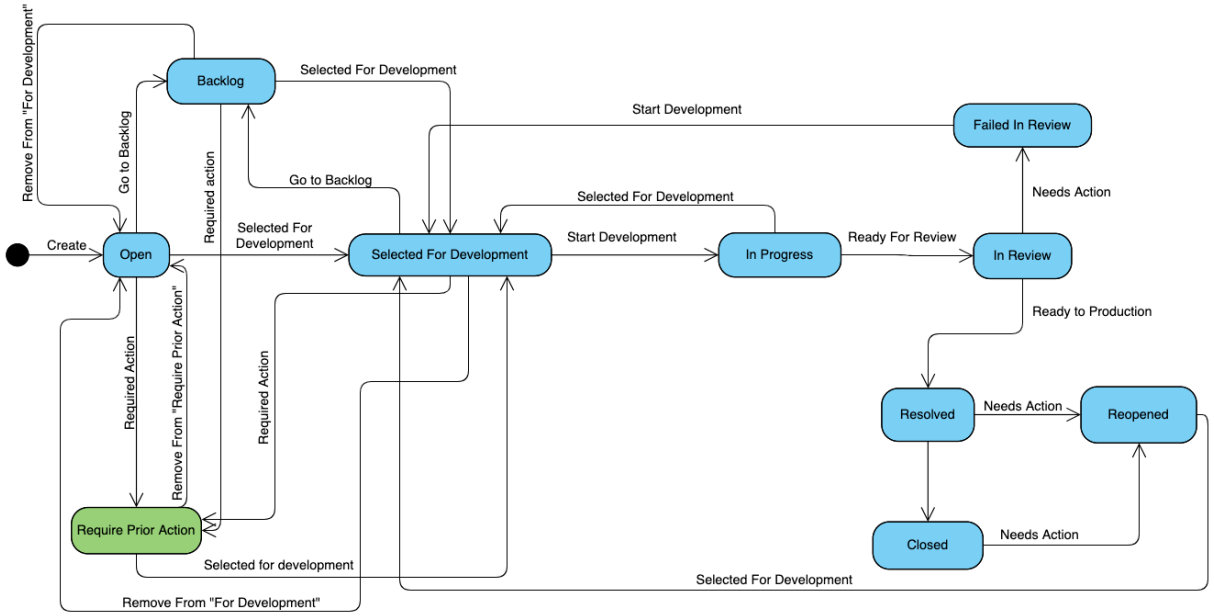
ANEXO H: Melhoria do Issue Creator




Bug Report
Improvement Request
Feature Request
Educational Content Request
What's an improvement!? An improvement makes doing something that was possible before simpler, more powerful or adds somehow to existing functionality.
GENERAL INFO
Summary *
Summary
A good summary should quickly and uniquely identify the desired behavior. Please, create a title by answering 3 questions: What is the desired behavior? in which feature? Where?
Bad example: Melhoramento no Track & Engage
Good example: Filtrar por Regex na definição de objectivos via URL do Track & Engage, na gestão dos goals
Component *
Choose the component related to the Improvement
The component is the main context of the issue. Basically you need to ask yourself: The issue relates with what!?
Please add the customer information you think is important to the Improvement Request you are making.
Add Element
Client ID
BEHAVIOR INFO & GOALS
Current Behavior *
Current Behavior
Describe in detail the way it works. Basically how is the workflow right now.
Desired Behavior *
Desired Behavior
Describe in detail how you would like the workflow to be.
Goal *
Goal
Describe in detail why do you think this improvement is important and the benefits that it will bring to the costumer.
Story Description:
Actualmente, {{CURRENT_BEHAVIOR}}, gostava, {{DESIRED_BEHAVIOR}}, para, {{GOAL}}
Additional Information / Suggestions / References
Additional Information / Debug Information
Do you have any additional information? Screenshots? Competitors information? Suggestions on how we should do it? etc?
CUSTOMER INFORMATION
Tell us more about the customer where the request happened, this is very important for us to define the issue priority.
Business Value *
Unknown / NA
What's the business value of the customer for us?
Customer Impact *
Unknown / NA
Tell us about the daily basis operational impact for the customer
Customer Satisfaction *
Unknown / NA
What's the current relationship status?
CREATE IMPROVEMENT REQUEST


Bug Report
Improvement Request
Feature Request
Educational Content Request
What's a feature!?
GENERAL INFO
Summary *
Summary
A good summary should quickly and uniquely identify the desired behavior. Please, create a title by answering 3 questions: What is the desired behavior? In which feature? Where?
Bad example: Download reports
Good example: Adicionar funcionalidade de download dos reports de e-mail.
Component *
Choose the component related to the Feature
The component is the main context of the issue. Basically you need to ask yourself: The issue relates with what!?
Please add the customer information you think is important to the Feature Request you are making.
Add Element
Client ID
BEHAVIOR INFO & GOALS
Desired Behavior *
Desired Behavior
Describe in detail how you would like the workflow to be.
Goal *
Goal
Describe in detail why do you think this improvement is important and the benefits that it will bring to the costumer.
Additional Information / Suggestions / References
Additional Information / Debug Information
Do you have any additional information? Screenshots? Competitors information? Suggestions on how we should do it? etc?
CUSTOMER INFORMATION
Tell us more about the customer where the request happened, this is very important for us to define the issue priority.
Business Value *
Unknown / NA
What's the business value of the customer for us?
Customer Impact *
Unknown / NA
Tell us about the daily basis operational impact for the customer
Customer Satisfaction *
Unknown / NA
What's the current relationship status?
CREATE FEATURE REQUEST

ANEXO I: Fluxo com novo Estado



ANEXO J: Issue Resolutions

Issue Type	Positive or negative	Issue Resolution	Description
Bug	Negative Resolution	Won't fix - Cannot reproduce	All attempts to reproduce the bug have failed, or there is not enough information to reproduce it. There is no evidence of this behavior happening. In case it continues to appear, please reopen the issue
		Won't fix - Duplicate bug	The declared problem is duplicated
		Won't fix - Not a bug	The current behavior is the supposed one. It can lead to an improvement
	Positive Resolution	Fixed - Solved with reserves	The bug was partially resolved for the specific case to prevent, however, it may happen again
		Fixed	A bug has been fixed
Improvement; New Feature; Task; Sub-Task	Negative Resolution	Won't do - already exists	The proposed functionality or improvement already exists
		Won't do - deprecated	The feature will not support any more improvements or new features. This may lead to a new approach.
	Positive Resolution	Done - Solved in another issue	It was solved in the resolution of another related issue
		Done	A feature or improvement is made