Mestrado em Informática e Sistemas

# Processo de Desenvolvimento de Software: Um caso de estudo de análise, implementação e validação

Relatório de estágio apresentado para a obtenção do grau de Mestre em Informática e Sistemas
Especialização em Engenharia de Software

**Autor**
**Filipe da Costa Rainho**

**Orientador**
**Jorge Barreiros**
Professor do Departamento de Engenharia Informática e Sistemas
Instituto Superior de Engenharia de Coimbra

**Supervisor**
**Cristóvão Cleto**
Crossing Answers

**Coimbra, Junho, 2019**

Este trabalho é dedicado à minha família que me deu o apoio necessário para concluir o mestrado.

# Abstract

The search for maturity of processes ensuring that they remain lightweight and requiring low levels of effort is a major challenge for software development organizations, and Crossing Answers is no exception. In recent years there has been a significant growth in the projects developed, both in number and in complexity, which has led to a growth in human resources. Due to this set of factors the processes began to be limited and inefficient for both project management and development itself. This is because the existing processes are neither structured or documented, giving the teams the ability to do the work ad hoc. In this way, it becomes much more difficult to guarantee the quality expected by the customers / users. This work tries to analyze the state in which the organization is at the level of processes in order to identify the main problems in them. In this way, a combination of the agile software development and the Capability Maturity Model Integration (CMMI) will be used and, later, provide solutions to the problems encountered. The evaluation is based on the Goal Question Metric (GQM) approach. After the evaluation and analysis carried out, it is verified that the great majority of the defined objectives were achieved, and the number of problems that the previous projects presented in this work have been greatly reduced. In this way, with improvements in the structured processes, training actions and workshops were carried out for all employees of the organization.

**Keywords:** process optimization; software development process; CMMI; Scrum; agile; GQM.

# Resumo

A procura pela maturidade dos processos garantindo que permanecem leves e que exigem baixos níveis de esforço é um grande desafio para as organizações de desenvolvimento de *software*, não sendo a Crossing Answers uma exceção. Nos últimos anos verificou-se um crescimento significativo nos projetos desenvolvidos, tanto em número como em complexidade o que levou a um crescimento ao nível dos recursos humanos. Devido a este conjunto de fatores os processos começaram a ser limitados e ineficientes tanto para a gestão de projetos como para o desenvolvimento em si. Isto porque os processos existentes não se encontram estruturados nem documentados, dando margem às equipas para realizar o trabalho de forma *ad hoc*. Desta forma, torna-se muito mais difícil garantir a qualidade esperada pelos clientes/utilizadores. Este trabalho procura analisar o estado em que a organização se encontra a nível de processos de forma a identificar os principais problemas existentes nos mesmos. Desta forma será usada uma combinação entre o desenvolvimento de *software* ágil e o Capability Maturity Model Integration (CMMI) e, posteriormente, fornecer soluções para os problemas encontrados sendo a avaliação realizada com base da abordagem Goal Question Metric (GQM). Após a avaliação e análise realizadas verifica-se que a grande maioria dos objetivos definidos foram alcançados, tendo sido diminuído em grande parte a quantidade de problemas que os projetos realizados anteriormente este trabalho apresentam. Desta forma, tendo sido verificadas melhorias nos processos estruturados foram realizadas ações de formação e workshops para todos os colaboradores da organização.

**Palavras Chave**: otimização de processos; processo de desenvolvimento de software; CMMI; Scrum; ágil; GQM.

# Agradecimentos

Primeiramente, gostaria de agradecer ao meu orientador, Professor Doutor Jorge Barreiros do Departamento de Engenharia Informática e Sistemas do Instituto Superior de Engenharia de Coimbra, que me acompanhou ao longo deste estágio de mestrado. Só tenho a agradecer a sua disponibilidade sempre que tive dúvidas e que me ajudou a seguir a direção correta ao longo dos últimos meses. Aos docentes das unidades curriculares que frequentei que me permitiram adquirir conhecimento para a realização deste trabalho.

Também gostaria de agradecer ao Cristóvão Cleto, responsável pelo meu estágio e ao Franco Vingadas por terem permitido a realização deste projeto na Crossing Answers, dando-me liberdade para realizar todas as experiências e atividades necessárias para validar o trabalho realizado. Além disso gostaria de agradecer a todos os meus colegas de trabalho da organização de acolhimento pelo apoio prestado durante todo o processo.

Serei sempre grato a todos os meus familiares, em especial aos meus pais, que me prestaram apoio ao longo dos meus estudos e especialmente durante a realização deste estágio de mestrado.

Por fim, quero deixar a minha gratidão e uma palavra de apreço aos meus amigos Francisco Couceiro, Rui Abreu, Tiago Guiomar e em especial ao Hugo Brito que me acompanhou ao longo destes dois anos nas unidades curriculares do mestrado e também durante a escrita deste trabalho.

# Índice

# Lista de Figuras

# Lista de Tabelas

# Definições e Acrónimos

**PDS** – *Processo(s) de Desenvolvimento de Software* – pode ser visto como um conjunto de atividades organizadas, usadas para definir, desenvolver, testar e manter um *software*.

**TI** – *Tecnologia da Informação* – conjunto de todas as atividades e soluções dotadas por recursos de computação que visam a produção, o armazenamento, o acesso, a segurança e o uso da informação.

**CMMI** – *Capability Maturity Model Integration* – modelo de referência que abrange atividades de desenvolvimento para produtos e serviços, contém várias práticas que dão suporte para a gestão de projeto e processos, engenharia de sistemas e de *software*.

**GQM** – *Goal Question Metric* – avaliação de processos em que a abordagem tem como baseia a definição de uma estrutura hierárquica de objetivos, questões e métricas

**PO** – *Product Owner* – papel do Scrum que tem como objetivo representar os *stakeholders* e o negócio.

**SM** – *Scrum Master* – papel do Scrum que garante a utilização dos processos, dos valores e das práticas do Scrum.

**ST** – *Scrum Team* – papel do Scrum que representa o grupo de colaboradores responsáveis pelo desenvolvimento do projeto.

**MVP** – *Minimum Viable Product* – é um produto com funcionalidades suficientes para satisfazer os primeiros utilizadores e fornecer *feedback* para desenvolvimentos futuros.

**REST** – *Representational State Transfer* – é um estilo de arquitetura de *software* que define um conjunto de restrições a serem usadas para a criação de *web services* usando o protocolo HTTP.

**API** – *Application Programming Interface* – é um conjunto de rotinas, protocolos e ferramentas para a criação de *software*, sendo a API a especificar como os diferentes componentes do *software* devem interagir.

**VoIP** – *Voice over Internet Protocol* – é a transmissão de voz e multimédia em redes IP (*Internet Protocol*).

# 1 Introdução

Atualmente, o desenvolvimento de *software* é caracterizado por mudanças rápidas nos objetivos e nos requisitos, o que requer agilidade por parte da organização, mas também maturidade para saber lidar com a mudança. Para tal, a adoção de processos ágeis, ajustados à realidade de cada empresa e capazes de dar resposta eficiente e eficaz a estas necessidades, é da maior importância.

Este relatório de estágio de mestrado descreve os trabalhos de análise e melhoria de Processos de Desenvolvimento de *Software* (PDS) numa empresa de acolhimento, a Crossing Answers, e foi realizado em 2018 como parte do mestrado em Informática e Sistemas - ramo de Desenvolvimento de Software do Departamento de Engenharia Informática e Sistemas.

## 1.1 Motivação

A organização de acolhimento na qual teve lugar este trabalho é a Crossing Answers, que é um estúdio de desenvolvimento digital com cerca de seis anos de existência ("Crossing - Digital development studio," n.d.). O desenvolvimento de *software* é o principal negócio da organização, sendo feito tanto para clientes como para as soluções internas existentes.

A Crossing Answers, nos últimos anos, teve um crescimento significativo nos projetos desenvolvidos, tanto em número como em complexidade. Este aumento levou a um crescimento ao nível dos recursos humanos. Desta forma os processos existentes nesta *startup* de base tecnológica começam a ser limitados e ineficientes tanto para a gestão de projetos como para o desenvolvimento em si. Isto porque os processos existentes não se encontram estruturados nem documentados, tornando difícil para as equipas ter regras no seu fluxo de trabalho. Desta forma, torna-se muito mais difícil garantir a qualidade esperada pelos clientes/utilizadores.

Após os responsáveis pelo departamento de Tecnologia da Informação (TI) terem percebido que se tinha chegado a uma situação muito pouco sustentável no que diz respeito à gestão e desenvolvimento dos projetos, foi necessário tomar medidas de forma a perceber os principais problemas que afetam a organização. Estas medidas foram:

1. Levantamento e avaliação ao fluxo de trabalho que existia na organização.
2. Levantamento e análise do histórico de retrospetivas dos projetos.
3. Recolha de *feedback* junto dos colaboradores.

Desta forma foi possível perceber que as principais áreas afetadas são o levantamento e gestão (já em formato de *Epics* e *User Stories* num *Product Backlog*) dos requisitos, a comunicação com o cliente tanto por questões de gestão do *Product Backlog* como por questões de revisão e recolha de *feedback* relativo a cada iteração, a inexistência de processos relativos aos testes, aceitação e validação do *software*

desenvolvido e a preparação e escolha de dependências externas. Estes fatores têm influenciado, diretamente, a qualidade do *software* desenvolvido e também a motivação dos colaboradores.

Como resultado desta avaliação, surgiu a necessidade de melhorar os processos de desenvolvimento adotados na empresa, o que deu origem ao trabalho descrito neste texto.

## 1.2 Objetivos e Metodologia de Trabalho

Este trabalho tem com principais objetivos:

1. Realização de um levantamento exaustivo dos processos existentes na organização no que diz respeito ao desenvolvimento de software.
2. Identificação dos principais problemas encontrados nos processos da organização
3. Estruturação dos processos que apresentaram mais problemas na organização.
4. Criação de documentação e *templates* para auxiliar o uso dos processos definidos.
5. Aplicação e avaliação do processo após a estruturação em projetos da organização e, no caso de terem existido melhorias, atualização do mesmo e formação aos colaboradores.

Em paralelo com os pontos anteriores, foi realizada a revisão de literatura de forma a perceber quais as melhores técnicas e ferramentas para a otimização e definição de processos de forma a produzir uma base para futuras otimizações aos mesmos. No capítulo 3 é possível consultar em mais detalhe a metodologia empregue neste trabalho.

De forma a procurar atingir os objetivos anteriormente descritos será usado um modelo de processo existente, o CMMI-DEV. Ao longo do manual CMMI-DEV são fornecidas várias indicações para as organizações que têm os seus processos assentes numa abordagem ágil. Para complementar essa informação será usado também o manual "A Guide to Scrum and CMMI: Improving Agile Performance with CMMI", produzido pelo CMMI Institute e orientado para organizações com os seus processos baseados em Scrum. Através da revisão feita ao manual (Dalton et al., 2016) foi encontrada uma listagem com diversos problemas comuns em organizações que desenvolvem *software*. Após uma análise feita a essa listagem, foram encontrados vários pontos em comum com a análise feita ao estado inicial da organização. Desta forma, e tendo em conta a análise feita ao estado inicial da organização, ficou definido em reunião interna que o problema a ser resolvido numa primeira otimização seria: "Os requisitos são vagos ou ficam em aberto". Nesta otimização são envolvidas as áreas de processo *Desenvolvimento de Requisitos* e *Gestão de Requisitos* do CMMI.

O CMMI disponibiliza vários componentes por área de interesse, no caso deste estudo é considerado o "CMMI for Development" ou "CMMI-DEV". O CMMI-DEV é um modelo de referência que abrange atividades de desenvolvimento para produtos e serviços, contém várias práticas que dão suporte para a gestão de projeto e processos, engenharia de sistemas e de *software*, entre outros (CMMI Product Team, 2010). O CMMI é um modelo de processo que define o que uma organização deve fazer para definir, compreender e promover comportamentos que levem a um melhor desempenho. Com os cinco níveis de maturidade e os três níveis de capacidade, o CMMI-DEV define as práticas mais importantes que precisam

de ser demonstradas para construir bons produtos e serviços, envolvendo tudo num modelo abrangente (Dalton et al., 2016).

Para validar se as alterações ao processo se estão a refletir em melhorias ou não, serão definidas um conjunto de métricas que permitem avaliar a eficiência das alterações realizadas na área de processo em questão. As métricas desenvolvidas têm como base a abordagem GQM. Esta abordagem baseia-se na definição de uma estrutura hierárquica de objetivos, questões e métricas. A estrutura começa com um objetivo que é refinado em questões e cada questão é refinada num conjunto de métricas (objetivas ou subjetivas) (Basili, Caldiera, & Rombach, 2002).

As principais contribuições do trabalho, na perspetiva da empresa e de vários dos seus departamentos e grupos são:

- Um formato de otimização de processos utilizando as práticas do CMMI em abordagens ágeis (capítulo 5).
- Estruturação e otimização dos processos de desenvolvimento e gestão de requisitos e documentação da primeira versão do PDS da organização (capítulo 5 e 6).
- Definição de método para avaliação das alterações realizadas aos processos da organização (capítulo 7).
- Formação aos colaboradores da organização de forma a conseguirem usar o processo criado neste trabalho (capítulo 8).

## 1.3 Estrutura do Relatório

O relatório está estruturado da seguinte forma. No capítulo 2 é apresentada a revisão de literatura realizada com base em artigos científicos e manuais relacionados ao tema. O capítulo 3 contem a metodologia empregue e os procedimentos utilizados para a realização trabalho. No capítulo 4 é exposta a informação da organização utilizada como caso de estudo neste trabalho, além disso é apresentada a situação inicial em que foi encontrada organização e feita a avaliação das áreas que apresentam mais problemas. Em seguida no capítulo 5 são selecionadas as áreas de processo a otimizar e é feito o mapeamento entre métodos ágeis os recursos fornecidos pelo CMMI de forma a resolver os problemas encontrados. No capítulo 6 é feita a estruturação do processo inicial com os resultados obtidos do mapeamento realizado entre o Scrum e o CMMI. No capítulo 7 de forma a perceber o impacto da estruturação idealizada é aplicado o estudo a projetos reais da empresa e feita a avaliação com base nos resultados obtidos. Para finalizar o ciclo, no capítulo 8 é realizada a atualização do processo existente com as novas estruturas propostas e realizadas ações de formação para os colaboradores. Por fim o capítulo 9 contem o resumo e as principais conclusões desta dissertação de mestrado.

# 2 Revisão de Literatura

Este capítulo apresenta a revisão de literatura realizada com base no estudo de manuais e artigos. Na secção 2.1 é feita uma abordagem às metodologias ágeis de desenvolvimento de *software* sendo referidos os seus princípios e as principais *frameworks* com base nos princípios ágeis, tal como o Scrum que também é descrito em seguida sendo apresentados os principais papeis, cerimónias e o seu ciclo de vida. Em seguida na secção 2.2 é abordada a engenharia de requisitos e os seus subprocessos fundamentais, dando principal importância à elicitação de requisitos. Na secção 2.3 é apresentada a importância e aplicabilidade dos CMMs e o impacto que tiveram na criação do CMMI. É explicado como é que o CMMI pode ser combinado com as metodologias ágeis e quais as vantagens que a combinação pode trazer em casos em que é pretendido definir e melhorar os processos. Por fim na secção 2.4 é apresentado o GQM que representa uma abordagem para a criação de sistemas de medição das otimizações realizadas aos processos.

## 2.1 Metodologias Ágeis

O Manifesto para o Desenvolvimento Ágil de Software foi publicado por um grupo de profissionais de software em 2001, e segundo os principais valores do Manifesto Ágil (Beck, 2001) enfatizam o seguinte:

- **Indivíduos e interações** mais do que processos e ferramentas.
- **Software funcional** mais do que documentação abrangente.
- **Colaboração com o cliente** mais do que negociação contratual.
- **Responder à mudança** mais do que seguir um plano.

Abordagens ágeis são um conjunto de valores e técnicas para gerir e entregar projetos. Enfatizam o desenvolvimento iterativo e incremental, o planeamento adaptativo, a resposta rápida e flexível à mudança e a estreita colaboração com o cliente para criar *software* funcional (Dalton et al., 2016). Existem várias *frameworks* para o desenvolvimento ágil de *software*, tais como o Scrum (Abrahamsson, Salo, Ronkainen, & Warsta, 2002) e o Extreme Programming, também conhecido como XP (Abrahamsson et al., 2002).

O Scrum é uma abordagem empírica baseada na flexibilidade, adaptabilidade e produtividade, deixando em aberto as técnicas, métodos e práticas específicas de desenvolvimento de software no processo de implementação, fornecendo uma estrutura de gestão de projetos que foca o desenvolvimento iterativo (Lucia & Qusef, 2010). Originalmente surgiu para a gestão de projetos de desenvolvimento de produtos, mas atualmente pode ser usado por equipas de manutenção ou de desenvolvimento de *software* no geral (Lina & Dan, 2012). Segundo (Cohn, 2014), com o conjunto de características enumeradas anteriormente, o Scrum permite manter o foco na entrega do maior valor de negócio no menor tempo possível, isto porque as necessidades do negócio é que devem determinar as prioridades do desenvolvimento do sistema.

Apesar de não prescrever práticas de engenharia, o Scrum tem um conjunto de cerimónias e papéis definidos (Lina & Dan, 2012). Os principais papeis do Scrum e as suas principais responsabilidades são:

- ***Product Owner* (PO)**: é quem representa os *stakeholders* e o negócio, desta forma é ele o responsável por manter e priorizar o *Product Backlog* consoante as necessidades desses mesmos *stakeholders*.
- ***Scrum Master* (SM)**: é quem garante a utilização dos processos e dos valores e práticas do Scrum. Funciona como um "escudo" para equipa garantindo assim a sua plena funcionalidade e produtividade.
- ***Scrum Team* (ST)**: grupo de colaboradores (tipicamente 5-9 elementos) com capacidade de auto-gestão e *cross-functional* sendo responsáveis pela análise, implementação, testes, etc.

O ciclo de desenvolvimento no Scrum é designado por *Sprint* (Lina & Dan, 2012), sendo este a unidade base de desenvolvimento no Scrum. Cada *Sprint* é *time-boxed* e tem a duração de 1-4 semanas sabendo que durante o seu período de execução nunca pode ser interrompido ou alterado (apenas em casos de extrema urgência e com uma justificação plausível). O seu principal objetivo é a criação de um incremento no sistema que pode originar uma potencial entrega.

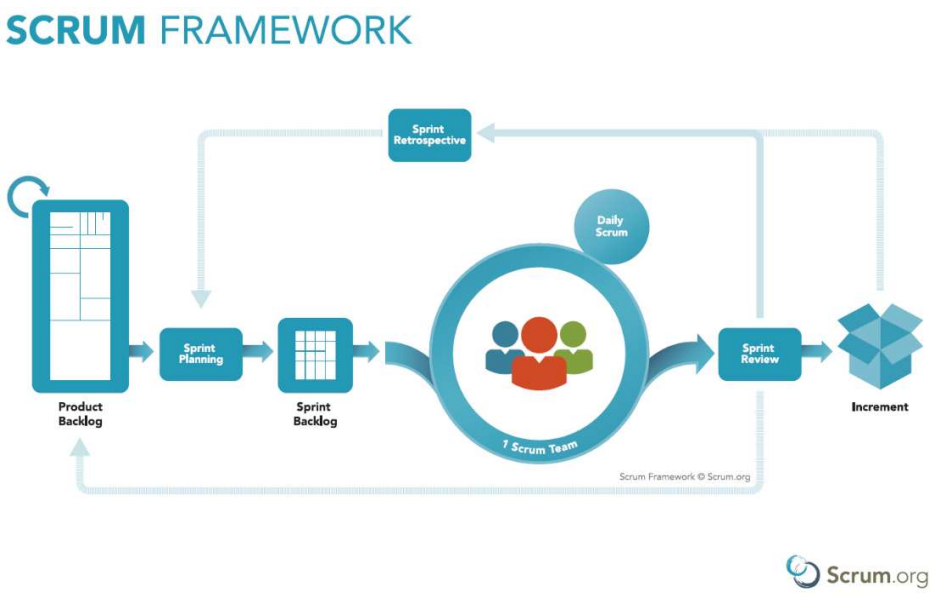



Figura 1 - Ciclo de vida do Scrum ("What is Scrum?," n.d.)

Tal como é apresentado na Figura 1, cada *Sprint* inicia com uma reunião de planeamento (*Sprint Planning*) onde a equipa seleciona do *Product Backlog* as tarefas com as quais se pode comprometer para o *Sprint* em questão. Antes de cada *Sprint Planning* o PO deve garantir que o *Product Backlog* se encontra devidamente atualizado e priorizado. Apesar do PO poder alterar, adicionar e remover itens do *Product Backlog* a qualquer altura, essas alterações apenas são consideradas no *Sprint Planning* seguinte. Após o final de cada *Sprint* é realizada a *Sprint Review* que tem como objetivo apresentar ao PO o trabalho realizado. Por fim, é realizada uma reunião onde é feita a retrospetiva (*Sprint Retrospective*) em que todos podem participar e devem responder as seguintes questões:

- O que fizemos bem?

- O que podíamos ter feito melhor?

O *Product Backlog* é uma lista priorizada de requisitos que o PO deseja implementar no sistema em desenvolvimento (Cohn, 2014). Deve ser atualizado e priorizado sempre que seja necessário dado que os requisitos devem ter valor para os *stakeholders*, no caso de surgirem novas necessidades devem ser usados os métodos de elicitação definidos pelo PDS. Os requisitos selecionados para o *Sprint Backlog* podem ser mais detalhados ou refinados caso exista a necessidade de mais clareza na sua especificação.

O Scrum promove a auto-gestão das equipas e incentiva a comunicação cara-a-cara entre todos os membros da equipa, uma das cerimónias existentes e que mais apela a esse fator é o *Daily Scrum Meeting* que consiste numa reunião com a duração de, no máximo, 15 minutos onde toda a equipa participa fazendo um ponto de situação relativamente ao seu trabalho diário. Um dos princípios fundamentais é o reconhecimento que durante um projeto o cliente pode mudar de ideias em relação ao que quer e precisa. De forma a dar resposta o Scrum adota uma abordagem empírica, aceitando que o problema pode não ser totalmente entendido ou definido no início, concentrando-se em maximizar a capacidade da equipa para entregar e responder aos requisitos mais emergentes (Lina & Dan, 2012).

As *startups* são empresas que exploram novas oportunidades de mercado e de negócio onde o ambiente é dinâmico, imprevisível e caótico devendo existir uma postura em que se age rápido, falha rápido e se aprende ainda mais rápido. Desta forma, é essencial existir a flexibilidade de acomodar mudanças, sendo as metodologias ágeis a solução mais viável (Giardino, Unterkalmsteiner, Paternoster, Gorschek, & Abrahamsson, 2014). É essencial diminuir o espaço de tempo no qual o cliente dá *feedback*. Segundo (Giardino et al., 2014) existem um conjunto de práticas comuns que as *startups* devem considerar, são elas:

- O uso de prototipagem evolutiva.
- A aceitação contínua do cliente.
- A entrega contínua com foco nas principais funcionalidades.
- O *empowerment* das equipas de forma a influenciar os resultados finais.
- O uso de métricas para aprender rapidamente com o *feedback* fornecido.
- O uso de ferramentas simples que facilitem o desenvolvimento e que permitam lidar com informação rapidamente.

## 2.2 Engenharia de Requisitos

Segundo (Ahrend, 2013), a engenharia de requisitos pode ser definida como o processo de descobrir o propósito da intenção de um *software*. Sendo o grau a que é atendido esse propósito a principal medida de sucesso. A fim de descobrir tal propósito, os *stakeholders* e as suas necessidades devem ser identificadas, documentadas, analisadas, comunicadas e posteriormente implementadas. Estas atividades são subdivididas em subprocessos, são eles:

- Elicitação de Requisitos.

- Análise de Requisitos.
- Gestão de Requisitos.
- Especificação de Requisitos.
- Validação de Requisitos.

Os requisitos são a base do design e arquitetura, sendo assim todos os projetos de desenvolvimento têm requisitos (CMMI Product Team, 2010) que podem ser de dois tipos, funcionais ou não-funcionais. Os requisitos funcionais representam funcionalidades concretas, capacidades ou serviços disponibilizados pelo sistema. Os requisitos não-funcionais são propriedades ou qualidades que os requisitos funcionais devem representar (Ahrend, 2013). Por vezes são referenciados como requisitos de qualidade, e a qualidade é sempre algo difícil de lidar na engenharia de requisitos. Alguns exemplos são: confiabilidade; usabilidade; eficiência; portabilidade; escalabilidade; privacidade; segurança; etc.

Resultados apontados em (Ahrend, 2013) revelam que o principal fator de fracasso de grandes projetos de *software* é devido à pouca precisão na elicitação de requisitos. Além disso é mais provável um software falhar devido à gestão deficiente dos requisitos, do que devido às más tecnologias, a prazos perdidos ou a problemas de gestão de alterações (Ahrend, 2013).

A elicitação de requisitos tem como objetivo identificar os *stakeholders*, os problemas subjacentes que precisam de ser abordados e, como resultado, identificar os limites e objetivos do sistema a desenvolver. O foco da elicitação de requisitos é puramente sobre os problemas e necessidades dos *stakeholders* e não sob o domínio da solução. Em (Ahrend, 2013) são enumeradas algumas das técnicas mais usadas na elicitação de requisitos, são elas: entrevistas, análise de documentação existente, *brainstorming*, *workshops*, prototipagem, métodos baseados em cenários e em objetivos, observação de participantes, entre outras. O maior desafio é selecionar o método mais apropriado e adequado para o projeto em questão, uma vez que a elicitação de requisitos é de especial interesse nas *startups*, devido ao seu foco na procura e validação das necessidades dos clientes. Desta forma as técnicas de elicitação são escolhidas com base na combinação de:

- Conhecimento de uma técnica em particular.
- A preferência do indivíduo.
- A aplicação de uma metodologia que suporta uma técnica em particular.
- Compreensão intuitiva de uma técnica adequada à circunstância atual.

Com base no estudo feito em (Lucia & Qusef, 2010), a maior diferença entre uma abordagem tradicional e uma abordagem ágil no que diz respeito à engenharia de requisitos não são os procedimentos que são feitos, mas sim quando são feitos. As abordagens tradicionais focam-se em recolher todos os requisitos e preparar todas as especificações antes de avançar para a fase seguinte, enquanto numa abordagem ágil a mudança nos requisitos é sempre recebida mesmo em fases finais do ciclo de desenvolvimento. Nas abordagens ágeis é aplicado foco nos valores mencionados no manifesto ágil, uma vez que o mesmo tem suporte muito eficiente no que diz respeito à engenharia de requisitos.

## 2.3 Modelos de Capacidade e Maturidade

Segundo (CMMI Product Team, 2010) os processos permitem alinhar a forma como o negócio é gerido, abordar a escalabilidade e fornecer uma forma de incorporar o conhecimento de como fazer melhor as coisas e alavancar os recursos e examinar as tendências do negócio. Os processos ajudam a força de trabalho de uma organização a atingir os objetivos, ajudando-os a trabalhar de maneira mais inteligente, e não mais difícil. Os processos eficazes fornecem um vínculo para introduzir e usar novas tecnologias de uma forma que melhor atenda aos objetivos da organização.

Os Capability Maturity Model (CMM) foram criados com o objetivo de ajudar as organizações a melhorar processos. Contêm elementos essenciais de processos eficazes para uma ou mais disciplinas e descrevem um caminho de melhoria evolutiva de processos *ad hoc* e imaturos a processos disciplinados e maduros com qualidade e eficácia melhorada (CMMI Product Team, 2010).

O CMMI surgiu para resolver o problema de existirem vários CMMs. Foi realizada a combinação dos modelos selecionados numa única estrutura de melhoria sendo feito o planeamento para o processo de melhoria em toda a organização. Para tal, na realização da combinação foram usados modelos que promovem consenso. Assim foi desenvolvido o primeiro modelo, o CMMI (versão 1.0.2) que foi projetado para o uso por organizações de desenvolvimento que procuram a melhoria de processos em toda a organização. O CMMI passou a "CMMI for Development" porque no momento do lançamento da versão 1.2 estava já planeado o lançamento de outros dois modelos ("CMMI for Acquisition" e o "CMMI for Services"). Por fim em 2010 foi lançada a versão 1.3 de cada um dos três modelos existentes, sendo nesse momento garantida a consistência entre os três (CMMI Product Team, 2010).

O CMMI é um modelo que consiste em descrições de práticas recomendadas para uma ampla gama de atividades de engenharia, abrangendo todo o ciclo de vida do produto desde a definição dos requisitos até à entrega e manutenção. Ajuda a integrar funções organizacionais tradicionalmente separadas, definir metas e prioridades de melhoria de processos, fornecer orientação para processos de qualidade e fornecer um ponto de referência para avaliar os processos atuais (Lina & Dan, 2012). Permite a melhoria de capacidade que fornece orientação para as organizações elevarem o desempenho, para tal fornece cinco níveis de maturidade em que cada um se baseia no anterior para promover a melhoria continua (Dalton et al., 2016). Os níveis são definidos da seguinte forma:

- *Initial* – os processos são geralmente *ad hoc* e caóticos.
- *Managed* – os projetos garantiram que os processos foram executados de acordo com a política definida.
- *Defined* – os processos são bem caracterizados e compreendidos e também são descritos em normas, procedimentos, ferramentas e métodos.
- *Quantitatively Managed* – são estabelecidos objetivos quantitativos para a qualidade e o desempenho do processo nos projetos da organização sendo posteriormente utilizados como critérios na gestão de projetos.

- *Optimizing* – os processos são melhorados continuamente com base numa compreensão quantitativa dos seus objetivos de negócio e necessidades de desempenho.

Uma assunção comum é que o CMMI é muito burocrático e adequado apenas para organizações maiores. Segundo (Lina & Dan, 2012) a assunção pode estar correta, mas pode ser bom tentar alcançar algumas das diretrizes do CMMI. Mesmo que não seja pretendido uma certificação logo à partida, poderão ser usadas partes dele para melhorar os processos existentes incluindo a sua qualidade.

Segundo (Lina & Dan, 2012) quando se estabelecem processos padrão para uso em todos os projetos da organização, deve ser garantido que:

- São implementados com sucesso em todos os projetos da organização.
- Periodicamente revisto com base nas retrospetivas.
- Compatível com práticas individuais e de equipa e flexível o suficiente para as equipas adaptarem o processo às suas necessidades.
- Documentado usando linguagem e formalismos que os profissionais entendam.

Deve existir um equilíbrio entre as responsabilidades individuais e organizacionais. Se o saldo for pesado de mais a favor das organizações, os colaboradores podem não ter a flexibilidade necessária e falharem em questões motivacionais. Por outro lado, muita flexibilidade pode expor a organização a riscos excessivos e com isso perder oportunidades de aprendizagem organizacional, que a médio longo prazo poderiam levar a uma melhor qualidade e produtividade. É difícil conseguir o equilíbrio certo (Lina & Dan, 2012).

Cada vez mais as organizações recorrem ao CMMI para melhorar o desempenho ágil. É usado para criar organizações escaláveis, resilientes e de alto desempenho e capacitar as organizações para compromissos de abordagens ágeis. Sendo o CMMI independente do ciclo de vida para qualquer metodologia de desenvolvimento (Dalton et al., 2016).

Ao nível do projeto, o CMMI tem foco num alto nível de abstração sobre o que os projetos fazem, e não em qual metodologia de desenvolvimento é usada, enquanto o Scrum tem foco em como os projetos são desenvolvidos. Portanto, segundo (Lina & Dan, 2012), o CMMI e o Scrum podem coexistir e complementar-se criando sinergias que beneficiam as organizações. O Scrum fornece instruções de desenvolvimento de *software* e o CMMI fornece as práticas de engenharia de sistemas, tais como práticas de gestão de projetos e suporte que podem ajudar a implementar, sustentar e melhorar a utilização do Scrum em pequenas e médias organizações.

No trabalho realizado em (Lina & Dan, 2012) é apresentada uma abordagem para combinar o Scrum e o CMMI em pequenas e médias empresas, sendo a abordagem a seguinte:

- Processos ao nível da organização – a maior parte dos benefícios ocorre quando o CMMI é implementado ao nível da organização, de modo a que todas as funções e capacidades que contribuem para o desenvolvimento sejam abordadas no processo de melhoria. Tal

como outras metodologias ágeis, o Scrum não chega ao nível da organização, desta forma os processos ao nível da organização deve seguir o CMMI.

- Atividades de gestão de projeto – o Scrum não fornece cobertura para todas as práticas da área de processo de gestão de projeto, mas pode ser ajustado para ser mais compatível com o CMMI. Neste caso são as práticas do Scrum que podem ajudar a complementar o CMMI.

O trabalho conclui que o Scrum é um ponto de partida recomendado para organizações com equipas pequenas e sem processos definidos, e que a sua combinação com o CMMI resulta, especialmente em pequenas e médias empresas. A sua combinação permite extrair benefícios de ambos e melhorar drasticamente o desempenho do negócio.

Segundo o trabalho realizado em (Selleri et al., 2015) é comum as organizações usarem metodologias ágeis como apoio para atingir o nível 2 e 3 do CMMI, sendo desta forma os seus esforços reduzidos. São destacados com principais benefícios da combinação do CMMI com metodologias ágeis os seguintes:

- Melhorias em aspetos organizacionais.
- Maior satisfação da equipa e do cliente.
- Mais fácil integração e assimilação dos processos.
- Reduções nos custos operacionais.
- Aumento da produtividade e redução nos defeitos.

É ainda frisado de que o treino, a visibilidade nos processos (comunicação cara-a-cara, *wikis*, manuais, *templates*, etc.) e as ferramentas para automatização de atividades complexas foram destacadas como fatores de sucesso. Apesar disso não existe uma solução genérica que atenda a todos os casos, as equipas e organizações devem basear-se nos seus próprios objetivos e concentrar-se nos seguintes fatores:

- A qualidade do processo manifesta-se através da satisfação da equipa e da organização com as práticas usadas para o desenvolvimento.
- A qualidade do produto manifesta-se através da satisfação do cliente com a obtenção de requisitos funcionais e não funcionais como desejado.

A versão 1.3 do CMMI-DEV incorpora sugestões para aplicação do modelo em metodologias ágeis, sendo essas sugestões também previstas no método de avaliação do CMMI, o SCAMPI (CMMI Product Team, 2010).

## 2.4 Goal Question Metric

O desenvolvimento de *software* requer mecanismos de medição de forma a obter *feedback* e avaliação. Esses mecanismos irão ajudar a organização em questões de planeamento de projetos, em determinar os pontos fortes e fracos dos processos atuais, em fornecer uma justificação para a adoção/melhoria de técnicas

e também para avaliar a qualidade dos processos. De acordo com o estudo realizado sobre a aplicação de métricas e modelos em ambientes industriais, a medição, para ser eficaz, deve ser:

- Focada em objetivos específicos.
- Aplicada a todos os processos do ciclo de vida.
- Interpretada com base na caracterização e compreensão do contexto organizacional, ambiente e objetivos.
- Definida de forma *top-down*, com base em objetivos e modelos.

No trabalho realizado em (Basili et al., 2002) é sugerida a abordagem GQM que é baseada no pressuposto de que para uma organização medir de forma intencional deve primeiro especificar os objetivos, em seguida traçar esses objetivos para dados que pretendem definir os objetivos operacionalmente e por fim fornecer uma estrutura para interpretar os dados em relação aos objetivos declarados. É afirmado que é importante deixar claro, pelo menos em termos gerais, quais as necessidades que a organização tem quando procede a um processo de otimização. O GQM fornece a especificação de um sistema de medição que visa um conjunto específico de questões e um conjunto específico de regras para a interpretação dos dados. O modelo resultante apresenta três níveis:

- Nível conceptual (*Goal*).
- Nível operacional (*Question*).
- Nível quantitativo (*Metric*).

No nível conceptual, um objetivo é definido para um objeto, por uma variedade de razões, com relação a vários modelos de qualidade, sob vários pontos de vista, em relação a um ambiente particular. Os objetos de medida podem ser:

- Produtos – artefactos, *deliverables* e documentos produzidos durante o clico de vida do sistema.
- Processos – atividades relacionadas ao *software* normalmente associadas ao tempo (especificar, testar, etc.).
- Recursos – itens usados por processos de forma a produzir *outputs*.

No nível operacional é usado um conjunto de perguntas para caracterizar a forma como a avaliação de um objetivo específico é realizada com base num modelo de caracterização. As perguntas tentam caracterizar o objeto de medição em relação ao problema de qualidade selecionado e determinar a sua qualidade a partir do ponto de vista selecionado.

No nível quantitativo, um conjunto de dados está associado a todas as perguntas para as responder de forma quantitativa. Os dados podem ser:

- Objetivos: se dependem apenas do objeto que está a ser medido e não do ponto de vista do qual eles são tomados (exemplo: nº de versões de um documento).

- Subjetivos: se dependem tando do objeto que está a ser medido como do ponto de vista do qual são medidas (exemplo: nível de satisfação do utilizador).

Na Figura 2 é apresentada a estrutura hierárquica do GQM. Tendo início num objetivo (*Goal*). Por sua vez é refinado em várias questões (*Question*). Por fim cada questão é refinada em métricas (*Metric*), algumas subjetivas e outras objetivas. Cada métrica pode ser usada em questões do mesmo objetivo ou então de objetivos diferentes.

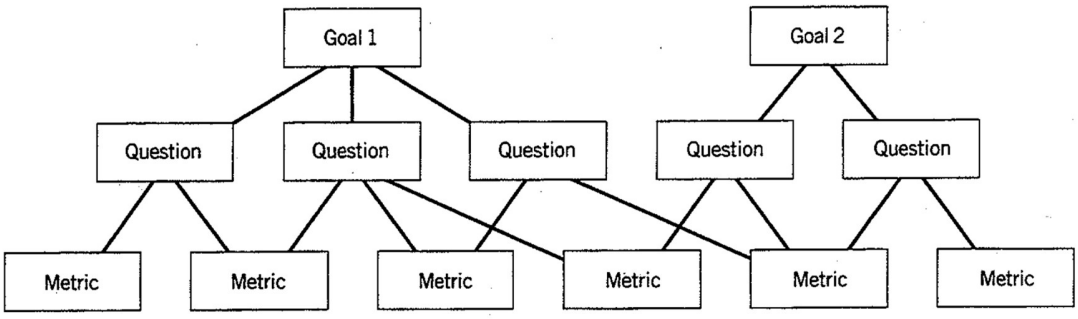



Figura 2 - Modelo GQM estrutura hierárquica (Basili et al., 2002)

Após a especificação anteriormente enumerada estar completa, é necessário desenvolver mecanismos de recolha, incluindo mecanismos de validação e análise. O processo de estabelecimento de objetivos é crítico para o sucesso da aplicação do GQM.

# 3 Metodologia

Este capítulo descreve como é que o trabalho foi realizado e qual a metodologia utilizada. Este trabalho teve como base dois grandes pilares, a avaliação do processo existente (secção 3.1) e, em seguida, a sua estruturação (secção 3.2). De forma a avaliar o processo reestruturado, este foi aplicado a um projeto real da empresa, sendo recolhidos e analisados os respetivos resultados de forma a verificar se a estruturação se refletiu numa melhoria efetiva (secção 3.3). Por fim, caso os resultados obtidos se apresentem como aspetos positivos o novo processo passa a ser utilizado por todas as equipas da organização, sendo também realizada a formação necessária para a sua utilização (secção 3.4).

## 3.1 Avaliação do Processo Inicial

A avaliação, feita com base num caso de estudo, tem por objetivo descobrir quais as principais áreas que poderiam ser melhoradas. Para tal foram realizadas as seguintes atividades:

1. Descrição da situação inicial da organização onde foi documentado todo o fluxo de trabalho desde a entrada de uma nova proposta até à entrega do projeto (secção 3.1.1).
2. Análise do histórico de retrospetivas de forma a perceber quais os pontos que as equipas apresentam como problemas de forma a melhorar esses aspetos em projetos futuros (secção 3.1.2).
3. Recolha de *feedback* juntos dos elementos que integraram as equipas dos projetos selecionados para a análise (secção 3.1.3).

### 3.1.1 Situação Inicial

Foi elaborado um documento com o objetivo de descrever todo o fluxo de trabalho usado inicialmente na organização desde a proposta comercial até à fase de suporte e manutenção. A sua necessidade prende-se no fato da organização não dispor de qualquer documentação relativa ao seu PDS. Desta forma foi possível ter uma perceção geral de como se desenrola todo o processo. O documento produzido apenas tem valor como critério de entrada para este trabalho e não como um anexo ou referencia a utilizar no futuro.

### 3.1.2 Retrospetivas do Histórico de Projetos

Nos projetos são realizadas retrospetivas no final de cada iteração (*Sprint*). Com isto pretende-se registar o *feedback* dos elementos das equipas de desenvolvimento, tanto de coisas que foram bem feitas como de coisas que podiam ter sido feitas melhor. Ao analisar este tipo de informação é possível perceber quais os pontos que apresentam um maior número de aparições e assim melhorar esses aspetos em projetos futuros. Para tal foram selecionados um conjunto de projetos, extraídas as retrospetivas registadas na ferramenta de

gestão de projetos e feita a devida análise das mesmas. O critério para a escolha dos projetos teve como base os seguintes fatores:

- Existirem retrospetivas realizadas pela equipa no decorrer do projeto.
- Existirem requisitos documentados na ferramenta de gestão de projeto.
- Ter dimensão suficiente em questões de tempo e de recursos humanos.

### *3.1.3* Recolha de *Feedback*

De forma complementar a informação das retrospetivas, foi também recolhido *feedback*, de forma geral, junto das equipas relativamente aos problemas que encontraram no fluxo de trabalho existente no decorrer dos projetos. O *feedback* fornecido foi dado por um ou mais elementos que integraram o projeto em questão. Foi usado o mesmo conjunto de projetos em que foram analisadas as retrospetivas. Toda a informação recolhida foi realizada num formato de entrevista, isto de forma totalmente informal, onde a agenda existente foi perceber o *feedback* dos colaboradores em relação ao fluxo de trabalho.

## 3.2 Estruturação do Processo

O processo inicial não se encontra documentado. É desejável que o processo esteja definido para que todos o conheçam e consigam consultar, ainda assim também deve ser configurável de forma a dar resposta a futuras necessidades das equipas. Desta forma o processo inicial deve ser estruturado de forma a permitir possíveis alterações no futuro. O processo estruturado teve como base um modelo existente, o CMMI, com o objetivo de melhorar/resolver o maior número de problemas encontrados na análise realizada. Também, e de forma a permitir realizar uma avaliação do processo após a aplicação do mesmo, foi utilizado o GQM para a definição de métricas que permitem avaliar a eficiência do processo relativamente às áreas em que se pretende realizar a otimização, neste caso relacionadas com o desenvolvimento e gestão de requisitos.

Por estruturação entende-se o conjunto de atividades que são necessárias realizar de forma a otimizar um processo, corrigindo o maior número de problemas possíveis no momento da estruturação. No caso deste trabalho foram usadas ferramentas para auxiliar a estruturação, são elas o CMMI e as abordagens ágeis de desenvolvimento de *software*.

O trabalho desenvolvido foi melhorado até chegar a uma primeira versão completa e pronta para aplicar em cenários reais.

## 3.3 Aplicação e Avaliação do Estudo Realizado

Para proceder à aplicação e avaliação do estudo foi criado um caso de estudo com um conjunto de projetos reais realizados na organização. A avaliação foi realizada para garantir que o processo estruturado realmente contribuí para a organização melhorar a qualidade dos processos e do *software* produzido.

Numa primeira fase, foi realizada a preparação do estudo. Foi feita a preparação das métricas, a criação dos inquéritos e o desenvolvimento de um *dashboard* simples onde é possível carregar a informação exportada da plataforma de gestão relativa a um determinado projeto. Posto isto foi então feita a aplicação

do estudo, para tal foi selecionado um projeto a ser realizado para um cliente e todo o seu desenvolvimento teve em conta as alterações ao processo sugeridas neste trabalho.

Terminado o desenvolvimento do projeto e a entrega ter sido efetuada foi realizado um estudo comparativo entre projetos realizados anteriormente e posteriormente a este trabalho. Por fim foram apresentados os resultados à gerência e realizado um debate relativo aos resultados obtidos.

## 3.4 Atualização do Processo e Formação

Após cada avaliação o processo existente deve ser atualizado caso se tenha chegado à conclusão de que a estruturação se tenha refletido em melhorias. De forma geral, a estruturação realizada neste trabalho revelou-se benéfica, sugerindo a sua integração no processo e a sua utilização nos restantes e futuros projetos da organização.

Além da atualização do processo é necessário dar a formação necessária aos colaboradores que vão trabalhar direta e indiretamente com o processo em questão. Sendo assim, foram realizadas dois *workshops* com o intuito de sensibilizar os colaboradores não só relativamente às atualizações feitas no processo como também no processo em geral e nas metodologias em que o mesmo se baseia.

# 4 Avaliação da Situação Inicial

Este capítulo apresenta a avaliação da situação inicial da entidade de acolhimento de forma a dar uma visão de como funciona e quais os problemas que existem no processo inicial.

Na secção 4.1 é feita a apresentação da entidade de acolhimento sendo descrita a sua estrutura e funcionamento inicial. Na secção 4.2 é explicado a forma como será feita a análise e quais as informações que serão recolhidas para proceder à mesma. Em seguida na secção 4.3 são apresentados os dados obtidos na recolha e na secção 4.4 é realizada uma análise alargada da informação obtida. Por fim na secção 4.5 são apresentadas as conclusões e ilações da análise realizada.

## 4.1 Entidade de Acolhimento

A Crossing Answers desenvolve vários tipos de projetos de *software* para clientes, além disso os seus colaboradores, sempre que necessário, auxiliam no desenvolvimento e suporte dos produtos comercializados pela organização. Um dos produtos desenvolvidos é o Luope, que consiste numa solução de *smart vending*, e que neste momento já conta com uma equipa praticamente independente. A equipa do Luope dá suporte a outros dois produtos, o Biller e o OneSW. O outro produto é o Andy, que neste momento ainda partilha alguns recursos com a Crossing Answers mas que tem como objetivo, num futuro próximo, formar uma *spin-off* tal como se sucedeu com o Luope.

No início do ano 2015 contava com uma equipa de três programadores que trabalhavam de forma não estruturada lidando com vários projetos e clientes. No início de 2017 a equipa de programadores quase triplicou e aí começaram a surgir um maior número de problemas. Aquando da realização deste trabalho a organização contava com 22 colaboradores incluindo o Luope e o Andy. As instalações são partilhadas, tal como os horários, atividades disponibilizadas (formações, *workshops*, *team building*, etc.) e também a grande maioria dos processos (mesmo sem que estejam ainda documentados).

Com o crescimento vivido nos últimos anos foi necessário mais estrutura e organização, especialmente no que diz respeito ao departamento de TI. Alguns dos fatores que tiveram mais impacto foram:

- A comunicação tanto interna como externa (por exemplo com o cliente).
- A adoção de um processo para o desenvolvimento dos projetos.
- A garantia de que os colaboradores consigam trabalhar em conjunto com o mesmo processo evitando que existam colaboradores com competências únicas.

Na Figura 3 pode ser consultado o organograma da organização. No topo da hierarquia está a direção executiva e a direção financeira, representadas pelo *Chief Executive Officer* (CEO) e pelo *Chief*

*Financial Officer* (CFO), respetivamente. Respondendo diretamente a este grupo está a direção operacional representada pelo *Chief Operating Officer* (COO) que é responsável pelas equipas de design e de TI, o departamento de Recursos Humanos e Logística, a direção comercial representada pelo *Chief Commercial Officer* (CCO) e por fim a direção de marketing representada pelo *Chief Marketing Officer* (CMO). Em paralelo ao TI existem as soluções internas comercializadas pela organização, onde cada uma delas tem um conjunto de grupos de trabalho semelhante aos grupos do TI (em certos casos podem ser partilhados consoante as necessidades num determinado momento). Este trabalho foi realizado no departamento de TI da Crossing Answers e está totalmente direcionado para o mesmo.

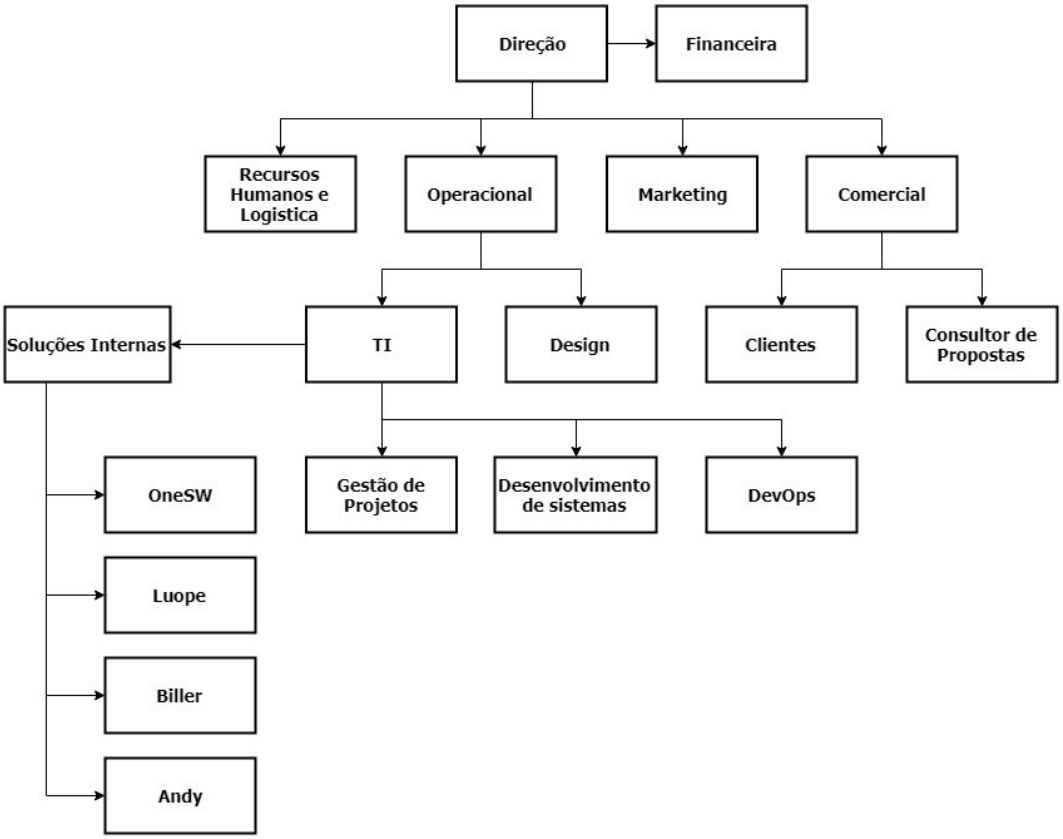



Figura 3 - Organograma Crossing Answers

Numa fase inicial existia uma grande acumulação dos cargos de gestão, o que se tornou impraticável manter de forma sustentável com o crescimento a que a organização foi sujeita. Desta forma foi necessário, por parte da gerência, criar uma hierarquia com mais níveis e delegar responsabilidades a outros colaboradores da organização para alguns dos cargos de gestão. Em alguns casos ainda é mantida a acumulação, isto porque a dimensão atual não justifica a delegação a um novo recurso a tempo inteiro.

Anteriormente à realização deste trabalho, existiu a adoção do Scrum como metodologia de desenvolvimento. Foram vários os motivos que levaram a optar pelo Scrum:

- Criação de um intervalo de tempo *timeboxed* onde o cliente não pode intervir.

- Possibilidade de o cliente priorizar os requisitos para as iterações seguintes.
- Possibilidade de o cliente alterar os requisitos para as iterações seguintes.
- Mais facilidade na gestão do requisitos e possibilidade de calendarização dos desenvolvimentos seguintes.
- Possibilidade da equipa de desenvolvimento se concentrar num determinado conjunto de tarefas durante um espaço de tempo definido.
- Permitir envolver mais o cliente durante o desenvolvimento, sendo atualizado sempre no final de cada iteração.
- Disponibilizar, por defeito, um conjunto de papeis com responsabilidades definidas.

Apesar de algumas das cerimónias e artefactos já terem sido adotados ainda não estão cimentadas nem são utilizadas bem o suficiente para ser um benefício para a organização. De forma a definir uma estrutura mais sólida e que permita uma melhor compreensão e adaptação das equipas, foram criados diferentes grupos para assumir as diferentes responsabilidades que existem ao longo do desenvolvimento de um projeto.

O departamento de TI é dividido em três grupos, cada um deles com as suas responsabilidades nos respetivos projetos que se encontram a trabalhar.

- **Gestão de projetos**: é um grupo que é responsável por gerir todo o processo de desenvolvimento, incluindo o levantamento e gestão dos requisitos com os *stakeholders*, a calendarização e a priorização e validação dos requisitos.
- **Desenvolvimento de sistemas**: é um grupo responsável pelo desenvolvimento dos projetos. Por norma são equipas pequenas, mas que trabalham sempre em Scrum ou Kanban (dependendo do projeto). Em cada equipa alocada ao projeto existe sempre um elemento responsável pela coordenação da equipa. São ainda responsáveis pelos testes e garantia da qualidade.
- **DevOps**: é um grupo responsável por providenciar todas as necessidades de infraestrutura de TI (servidores, redes, servidores de bases de dados, preparação dos ambientes de *deployment* tanto de desenvolvimento, como de aceitação e também de produção) que vão surgindo em algum ponto do desenvolvimento do projeto e/ou na entrega.

Existe a política de tentar manter a hierarquia o mais horizontal possível, dentro do que pode ser sustentável para a organização. Isto porque a gerência tentar estar envolvida ao máximo nos projetos para dar suporte às equipas. Contudo nem sempre existe uma visão geral dos objetivos dos projetos quando são iniciados, nem uma boa definição do que realmente é para fazer. Esse conjunto de fatores tende a influenciar os tempos de desenvolvimento e a qualidade do projeto quando é entregue ao cliente. Com o passar do tempo o departamento de TI tem-se concentrado no desenvolvimento e não teve a possibilidade de, em conjunto com a gerência, alocar tempo para definir e documentar processos. Este trabalho pretende colmatar essa lacuna.

Quando este trabalho foi iniciado as equipas de desenvolvimento eram responsáveis por todo o processo de levantamento, especificação e gestão de requisitos. Além desta prática não estar de acordo com a metodologia que se pretende adotar (o Scrum), a forma como as atividades de levantamento, especificação e gestão dos requisitos não estão de acordo com qualquer padrão e cabe ao colaborador em questão decidir como especificar e documentar os mesmos já que não existe material de apoio.

A Crossing Answers não tem os seus processos devidamente definidos e documentados, um deles é o PDS. As equipas do departamento de TI seguem um conjunto de procedimentos que, atualmente, lhes é explicado apenas verbalmente. A grande maioria não consegue seguir a sua totalidade, especialmente os procedimentos e tarefas mais relacionados com levantamento e definição dos artefactos necessários ao projeto.

Existe um conjunto de procedimentos executados pelos colaboradores dos diferentes departamentos da Crossing Answers. Apesar do conhecimento da sua existência e do uso dos mesmos como fluxo de trabalho, não existe qualquer documentação ou institucionalização dos mesmos. Desta forma é necessário proceder à documentação do PDS garantindo clareza para todos os colaboradores. Os procedimentos existentes inicialmente podem ser consultados em mais detalhe no Anexo A, tendo sido realizada a sua elaboração no âmbito deste trabalho de forma a perceber o fluxo de trabalho existente aquando do início da realização deste trabalho.

## 4.2 Preparação da Análise

De forma a atingir estes objetivos, irá ser analisada a informação disponível nas retrospetivas do histórico de projetos e o *feedback* recolhido junto dos colaboradores em relação ao processo. É importante aprender com os erros cometidos, e as retrospetivas são, sem dúvida, uma ótima forma de recolher essa informação já que a informação das retrospetivas permite saber quais as dificuldades que as equipas atravessaram, em curtos espaços de tempo, ao longo do desenvolvimento.

As retrospetivas, tal como a recolha de *feedback* diretamente com os colaboradores, podem ser usadas para impulsionar a mudança e fortalecer os processos. Desta forma será prestada especial atenção aos pontos que foram referidos como possibilidade de melhoria ou como problemáticos durante a execução dos projetos. Serão consideradas apenas as retrospetivas dos projetos selecionados para o estudo realizado neste trabalho.

## 4.3 Apresentação dos Resultados Obtidos

Nos projetos selecionados para este estudo foram realizadas retrospetivas no final das iterações pelos elementos das equipas relativas a cada projeto. A Tabela 1 apresenta o agregado dos itens respondidos na pergunta “O que devíamos ter feito melhor?” de todas as retrospetivas realizadas.

Tabela 1 - Revisão de retrospetivas

| ***Código*** | ***Retrospetiva*** | ***Ocorrências*** |
|---|---|---|
| ***R-1*** | Melhorar a comunicação e a recolha de *feedback* com o cliente | 5 |
| ***R-2*** | Mais cuidado e atenção no cumprimento e definição das datas de entrega | 1 |
| ***R-3*** | Melhorar planeamento das alocações, continuam a existir vários sprints interrompidos | 4 |
| ***R-4*** | Mais consideração pela velocidade das equipas de forma a que os Sprints acabem sem necessidade de *overworking* | 2 |
| ***R-5*** | Maior cuidado na definição da arquitetura, especialmente em projetos onde não exista grande à-vontade | 1 |
| ***R-6*** | Melhorar a forma como é feita a escolha das ferramentas necessárias aos projetos | 2 |
| ***R-7*** | Quando se trabalha com novas tecnologias deve existir um período de investigação antes de avançar para o planeamento | 1 |
| ***R-8*** | Quando um projeto requer informação de terceiros, essa mesma informação deve ser recolhida e fornecida á equipa antes do arranque do projeto | 2 |
| ***R-9*** | Melhorar o processo de estimativa das *User Stories* | 4 |
| ***R-10*** | As *User Stories* estão a ser mal definidas e pouco claras o que influência nas estimativas e no trabalho produzido | 3 |
| ***R-11*** | Deve existir mais clareza em cada *User Story* e também em cada *Epic* | 3 |
| ***R-12*** | Melhorar a forma como as *User Stories* são definidas | 4 |
| ***R-13*** | Melhorar a forma como são divididas as *Epics* de forma a ter um melhor controlo dos requisitos | 1 |
| ***R-14*** | Melhorar a definição das *Epics* de forma a criar todas as *User Stories* necessárias | 1 |
| ***R-15*** | Devem ser executados mais testes ao software desenvolvido | 4 |
| ***R-16*** | Deve existir mais clareza e regra na forma como é feita a aceitação do software | 4 |
| ***R-17*** | O *deployment* dos sistemas deve ser melhorado, tornando-o o mais automático possível | 2 |
| ***R-18*** | Deve existir um maior cuidado no versionamento do código | 1 |
| ***R-19*** | Os *layouts* são incompletos, devem prever todas as funcionalidades do sistema | 1 |
| *Total de tópicos das retrospetivas* | | 46 |

Como pode ser visualizado na Tabela 1 existem várias áreas afetadas, sendo o levantamento e gestão de requisitos a área com mais problemas, segundo as retrospetivas das equipas. Foi registado várias vezes que os requisitos são mal definidos ou que não tem clareza suficiente. Além disso são também registados vários problemas associados ás estimativas (que podem ser proporcionados pela pouca clareza dos requisitos) e à realização de testes.

O departamento de TI cresceu bastante nos últimos anos. Se incluirmos o grupo das soluções internas, a empresa conta, neste momento, com catorze colaboradores. Quando confrontados com a recolha de *feedback* em relação às dificuldades e problemas com que se deparam atualmente, a grande maioria enumerou os seguintes problemas:

- Os objetivos não são bem definidos no início do projeto.
- A informação que chega as equipas é, na grande maioria dos casos, insuficiente.

- A comunicação com o cliente e a obtenção de *feedback* por vezes é ignorada o que leva longos períodos de aceitação e de correções.

Ainda segundo o *feedback* obtido dos colaboradores, foram também recolhidas as seguintes sugestões de melhoria ao processo existente aquando o início deste trabalho:

- Definição de um processo de levantamento e gestão de requisitos para toda a organização que permita fazer chegar ás equipas informação útil.
- Criar modelos para escrever os requisitos e as suas especificações.
- Anexar o máximo de informação possível aos requisitos de forma a perceber a sua origem.
- Definir como será feita a comunicação com o cliente no início de cada projeto.
- Definição de processos para testes funcionais e de aceitação que permita testar as funcionalidades do projeto antes de fazer a entrega ao cliente.
- Definição de um processo para gestão de riscos.
- Melhorar a gestão de alocações e calendarização de forma a não criar compromissos irrealistas com os clientes. O tempo necessário para reuniões, *deployment*, manutenção e revisões deve ser considerado na calendarização.
- Prever a alocação de tempo em revisões de código e arquitetura, sendo necessário definir as convenções a utilizar na organização.

De forma geral, os colaborados concluíram a sua retrospetiva afirmando que não conhecem bem os processos e que não sabem como aplicar os mesmos em diferentes contextos.

## 4.4 Análise da Situação Inicial

Após reunir e agregar as retrospetivas realizadas pelas equipas nos projetos é possível verificar que existem vários problemas que necessitam de uma resolução. Tendo em conta a Tabela 1, foi possível agregar os problemas nos seguintes grupos:

- Comunicação com o cliente (R1).
- Calendarização e estimativas (R2; R9).
- Planeamento dos *Sprints* (R3; R4).
- Escolha e preparação da utilização de dependências externas (R5; R6; R7; R8).
- Levantamento e gestão de requisitos (R10; R11; R12; R13; R14).
- Testes e aceitação do *software* desenvolvido (R15; R16).
- *Deployment* e versionamento (R17; R18).
- Nível de detalhe dos *layouts* (R19).

Tendo em conta os resultados obtidos e a agregação realizada pode ser verificado que em cerca de 26% dos tópicos existentes nas retrospetivas realizadas são relados problemas associados ao levantamento

e gestão dos requisitos. O segundo grupo com mais percentagem de presenças nas retrospetivas é o grupo dos testes e aceitação do *software* desenvolvido com cerca de 17% dos tópicos existentes nas retrospetivas.

Relativamente ao *feedback* recolhido junto dos colaboradores, os problemas relatados dificultam o processo de levantamento e gestão de requisitos, como tal a grande maioria dos requisitos são muito pouco claros, em alguns casos resumem-se apenas a um título/designação. Com a falta de clareza nos requisitos as equipas sentem dificuldade em realizar estimativas e também em saber quais os critérios de conclusão de um determinado requisito. Existem outros problemas ligados aos requisitos, a quase inexistência de rastreabilidade nos requisitos que dificulta, tanto para as equipas de desenvolvimento como para a gestão de projetos, entender o que já foi ou não especificado, qual o seu estado e se existir qual a sua origem.

## 4.5 Conclusões

Tendo em consideração os resultados apresentados na secção 4.3 e analisados na secção 4.4, verifica-se que os principais pontos de melhoria devem incidir sobre o desenvolvimento e gestão dos requisitos. Apesar de terem sido encontrados outros problemas no fluxo de trabalho diário das equipas de TI, não será possível, no âmbito deste trabalho, proceder otimização de todos os pontos que necessitam de melhorias.

Foi escolhida uma área para proceder à otimização. Tendo em conta o elevado número de vezes que questões ligadas aos requisitos foram referidas nas retrospetivas e também na influência que os mesmos têm no sucesso de um projeto, foi escolhido como base para este trabalho as áreas ligadas à gestão e desenvolvimento de requisitos.

# 5 Requisitos no CMMI

Este capítulo tem como objetivo apresentar e mapear com métodos ágeis os recursos fornecidos pelo CMMI de forma a resolver os problemas encontrados no capítulo 4 na gestão e desenvolvimento de requisitos. Para tal na secção 5.1 é descrito o papel da engenharia de requisitos no desenvolvimento de *software* e são também descritas as duas áreas de processo do CMMI que irão ser estudadas neste trabalho, a área de processo *Desenvolvimento de Requisitos* e a área de processo *Gestão de Requisitos*. Na secção 5.2 é feito o mapeamento das áreas de processo com métodos ágeis de desenvolvimento de software. Para cada prática específica são apresentadas as vantagens da sua aplicação e também como pode ser satisfeita e demonstrada recorrendo a metodologias ágeis. Por fim na secção 5.3 é feita a análise e conclusões do mapeamento apresentado na secção anterior.

## 5.1 Introdução

Conforme determinado no capítulo 4 foi escolhido como base para este trabalho as áreas ligadas à gestão e desenvolvimento de requisitos. Tendo em conta a documentação fornecida pelo CMMI, após a sua consulta em (Dalton et al., 2016) foi encontrada uma listagem de vários problemas comuns às organizações que usam abordagens ágeis no desenvolvimento de *software*. Após uma análise à referida listagem foi encontrado um item nessa listagem que se identifica com alguns dos problemas relatados pelas equipas nas retrospetivas, é ele "Os requisitos são vagos ou ficam em aberto". O CMMI prevê a resolução deste problema através da adoção das práticas sugeridas nas áreas de processo *Desenvolvimento de Requisitos* e *Gestão de Requisitos*, em conjunto com as seguintes técnicas e cerimónias do Scrum:

- *Epic*/*User Story*.
- *Definition of Done*.
- *Backlog Grooming*.

Posto isto todo o trabalho realizado teve com base as áreas de processo, técnicas e cerimónias enumeradas anteriormente. Apesar disso foi possível chegar à conclusão que para fazer uma boa definição do processo de requisitos também seria necessário definir algumas etapas do PDS. Desta forma foi documentada a primeira versão do PDS da Crossing Answers Anexo B, anexo este criado no contexto deste trabalho. A sua criação teve uma grande importância para a organização, isto porque permitiu criar uma base a implementação de melhoria contínua com base nas retrospetivas e também para futuras otimizações de maior impacto nos procedimentos principais. Ainda que necessite de muito trabalho e algumas melhorias já permite a todas as equipas seguirem a mesma linha orientadora no desenvolvimento dos projetos.

### 5.1.1 Desenvolvimento de Requisitos

A área de processo *Desenvolvimento de Requisitos* tem como principal objetivo elicitar, analisar e estabelecer requisitos de clientes, de produtos e componentes de produtos. Em conjunto, os três tipos atendem às necessidades dos *stakeholders* e dos atributos do produto, abordando ainda as decisões tomadas em questões de design e arquitetura (CMMI Product Team, 2010). O desenvolvimento de requisitos inclui as seguintes atividades:

- Elicitação, análise, validação e comunicação das necessidades, expectativas e restrições do cliente para obter requisitos prioritários dos clientes que irão satisfazer os *stakeholders*.
- Recolha e coordenação das necessidades dos *stakeholders*.
- Desenvolvimento do ciclo de vida dos requisitos.
- Estabelecimento dos requisitos funcionais e de qualidade.
- Estabelecimento de requisitos de produtos consistentes com os requisitos do cliente.

Os requisitos são identificados e refinados ao longo das fases do ciclo de vida do produto. Decisões de design, ações corretivas e *feedback* durante cada ciclo são analisados quanto ao impacto nos requisitos. Para tal existem três objetivos específicos na área de processo de *Desenvolvimento de Requisitos*:

1. **Desenvolver os requisitos do cliente**: aborda um conjunto de requisitos do cliente para usar no desenvolvimento de requisitos do produto.
2. **Desenvolver os requisitos do produto**: aborda a definição de um conjunto de requisitos de produto a serem usados no design e arquitetura do produto.
3. **Analisar e validar os requisitos**: aborda a análise de requisitos do cliente e do produto para definir e entender os requisitos.

As análises são usadas para entender, definir e selecionar os requisitos e destinam-se a auxiliar as restantes práticas da área de processo.

### 5.1.2 Gestão de Requisitos

A área de processo de *Gestão de Requisitos* tem como principal objetivo gerir os requisitos dos projetos e garantir o alinhamento entre esses requisitos e os planos do projeto, para tal são geridos todos os requisitos recolhidos ou gerados durante o desenvolvimento do projeto. São tomadas medidas apropriadas para assegurar que os requisitos aprovados sejam geridos de forma a suportar as necessidades de planeamento e de execução do projeto (CMMI Product Team, 2010).

Quando um projeto recebe requisitos, esses requisitos são revistos de forma a resolver o máximo de problemas e evitar mal-entendidos antes que os requisitos sejam incluídos no plano do projeto. Após a revisão, deve ser criado um compromisso por parte dos intervenientes no projeto em relação ao que ficou definido.

Relativamente às mudanças, devem ser geridas à medida que os requisitos evoluem e que são identificadas inconsistências. Parte do trabalho da gestão de requisitos é documentar as mudanças de requisitos e as suas razões, mantendo rastreabilidade bidirecional com os requisitos originais.

Todos os projetos possuem requisitos. Nos casos de atividades de manutenção, as alterações são baseadas em alterações a requisitos, ao design e arquitetura ou à implementação. O mesmo se aplica a incrementos ao produto ou disponibilização de novos requisitos, isto porque a mudança também pode surgir devido à evolução das necessidades do cliente, manutenção, mudança de tecnologias e evolução dos padrões. Todas as alterações nos requisitos podem ser documentadas em pedidos de mudança ou assumir a forma de novos requisitos. Independentemente da sua origem ou formato, devem ser geridos de acordo como o processo.

Na área de processo de *Gestão de Requisitos* existe um único objetivo específico, chamado gestão de requisitos. Aborda a gestão dos requisitos e a identificação de inconsistências com o plano traçado para o projeto. Deve ser mantido um conjunto de requisitos atual e aprovado ao longo do ciclo de vida do projeto, para tal deve ser feito o seguinte:

- Gerir todas as mudanças nos requisitos.
- Manter a relação entre os requisitos, o plano do projeto e os produtos de trabalho.
- Garantir o alinhamento entre os requisitos, o plano do projeto e os produtos de trabalho.
- Tomar ações corretivas.

## 5.2 Mapeamento de Áreas de Processo do CMMI em métodos ágeis

Pretende-se usar as práticas presentes no modelo CMMI, para tal é feito um mapeamento dessas práticas de forma a perceber como o Scrum pode implementar e melhorar com a utilização do CMMI. Toda a informação apresentada tem como base o trabalho realizado em (Dalton et al., 2016).

### 5.2.1 Desenvolvimento de Requisitos

Em seguida é detalhado, para cada uma das práticas específicas da área de processo *Desenvolvimento de Requisitos*, como pode fortalecer o ágil, como o ágil pode satisfazer a prática específica e também como a sua aplicação pode ser demonstrada.

#### 5.2.1.1 SP 1.1 – Elicitação de necessidades

Esta prática dá ênfase à elicitação proativa das necessidades do negócio, clientes, equipa, entre outros. Garante que a criação de *Epics* é feita no início do ciclo de vida do produto em parceria com os *stakeholders*.

A prática pode ser satisfeita com a realização contínua de sessões de *Backlog Grooming*, de forma a identificar novos requisitos e mudanças nos existentes. Para tal o PO deve fornecer e apresentar as *Epics* a serem refinadas em *User Stories*, para que em conjunto com a equipa, procedam à sua documentação no *Product Backlog*.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Garantir a documentação do *Product Backlog* com a presença de *Epics* e de *User Stories*.

- Mapear *Epics* e *User Stories*, referenciando a *Epic* em cada *User Story*.
- O *Product Backlog* deve ser gerido numa ferramenta de gestão de projeto ou num quadro físico para esse propósito.

#### 5.2.1.2 SP 1.2 – Transformar as necessidades dos *stakeholders* em requisitos do cliente

Esta prática suporta a criação de *Epics* de *User Stories*, incluindo a priorização no *Product Backlog* que permite definir as restrições de implementação.

Pode ser satisfeita refinando as *Epics* em *User Stories* menores. O processo de refinação das *Epics* pressupõe a criação, alteração ou exclusão das *User Stories*. Deve ser realizado durante as sessões de *Backlog Grooming* que por sua vez devem decorrer com regularidade.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Garantir a documentação do *Product Backlog* com a presença de *Epics* e de *User Stories*.
- O *Product Backlog* deve ser gerido numa ferramenta de gestão de projeto ou num quadro físico para esse propósito.
- Garantir o mapeamento visual da hierarquia entre *Epics* e *User Stories*.

#### 5.2.1.3 SP 2.1 – Estabelecer os requisitos de produto e de componentes de produto

Esta prática fornece contexto para a arquitetura, interface e requisitos não-funcionais, sendo os mesmo recolhidos no formato de *User Story*.

Esta prática pode ser satisfeita respeitando o princípio de que a modificação de *Epics* e *User Stories* devido a alterações não obriga à exclusão das existentes. Podem ser criadas novas *User Stories* relacionadas à modificação de forma a contemplar o que foi alterado.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- O *Product Backlog* deve ser gerido numa ferramenta de gestão de projeto ou num quadro físico para esse propósito.

#### 5.2.1.4 SP 2.2 – Alocar requisitos dos componentes de produto

Esta prática garante que os requisitos são alocados a um incremento para posteriormente serem entregues. A alocação é realizada durante o *Sprint Planning*.

Para satisfazer esta prática as *User Stories* são dividas em tarefas que permitem definir a funcionalidade desejada no produto. É também definido para cada *User Story* uma *Definition of Done*.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- As *User Stories* devem conter tarefas que definem a sua funcionalidade.
- O *Product Backlog* deve ser gerido numa ferramenta de gestão de projeto ou num quadro físico para esse propósito.

#### 5.2.1.5 SP 2.3 – Identificar os requisitos de interface

Esta prática suporta a necessidade de recolher os requisitos de interface em formato de *User Story*. Desta forma para ver esta prática satisfeita é necessário identificar e desenvolver os requisitos de interface por meio de *User Stories*. Durante a refinação das *Epics* podem ser encontrados requisitos de interface adicionais.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Documentar requisitos de interface como User Stories no Product Backlog.

#### 5.2.1.6 SP 3.1 – Estabelecer conceitos e cenários operacionais

Esta prática promove o suporte da definição de cenários de negócio nas *User Stories*. O cenário de negócio deve descrever como a *User Story* deve ser implementada e qual a necessidade da sua implementação. Ambos os fatores devem ser evidentes.

Para satisfazer esta prática, durante a criação de *User Stories* no *Backlog Grooming* deve ser esclarecido e documentado o comportamento do requisito em questão.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Garantir a documentação do Product Backlog com a presença de *Epics* e de *User Stories*.
- As User Stories devem usar um formato que articula claramente o cenário de negócio com a forma como será realmente implementada.

#### 5.2.1.7 SP 3.2 – Estabelecer definições de funcionamento necessário e atributos de qualidade

Esta prática pretende garantir que é conseguido tudo o que é necessário para a disponibilização de um sistema totalmente funcional.

Para satisfazer esta prática, durante o *Backlog Grooming*, as *User Stories* são preenchidas com critérios de aceitação e atributos de qualidade que posteriormente poderão ser usados para questões de aceitação nas demonstrações realizadas para os *stakeholders*.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- As *User Stories* devem ter critérios de aceitação.
- Os scripts de testes implementam os critérios de aceitação.
- Na *Sprint Review* é realizada a validação se os critérios de aceitação são ou não atendidos.

#### 5.2.1.8 SP 3.3 – Analisar requisitos

Esta prática pretende garantir que as *User Stories* são realmente necessárias e que dão suporte ás *Epics*. Para tal é requerido que a rastreabilidade entre os requisitos seja suportada no processo.

Para satisfazer esta prática durante as sessões de *Backlog Grooming* o PO e a equipa analisam as *Epics* e as *User Stories* fazendo perguntas de esclarecimento para entender o que é necessário implementar. Deve ser criado para cada *User Story* uma *Definition of Done* com detalhe suficiente.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- As *Epics* e *User Stories* são atualizadas consoante o resultado da análise.
- A análise deve ser realizada durante as sessões de *Backlog Grooming*.
- Devem ser definidos os casos de teste iniciais e a *Definition of Done*.

#### 5.2.1.9 SP 3.4 – Analisar os requisitos para alcançar o equilíbrio

Esta prática pretende garantir o equilíbrio das *User Stories* no que diz respeito à prioridade, tamanho e esforço. Ajuda também na promoção da criação de "*spikes*", ou *User Stories* experimentais que são focadas em prototipagem, simulações ou provas de conceito.

Pode ser satisfeita com uma correta priorização das *User Stories*, deixando as mais prioritárias no topo. Esta prática é uma entrada para uma correta realização do *Sprint Planning* de acordo com as necessidades do negócio.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- As *Epics* e *User Stories* são documentadas e priorizadas da mais prioritária para a menos prioritária no *Product Backlog*.

#### 5.2.1.10 SP 3.5 - Validar requisitos

Esta prática pretende dar garantias de que as *Epics* e *User Stories* atendem as necessidades do cliente, sendo este um dos princípios do *Backlog Grooming*. Para proceder à validação podem ser utilizadas "*spikes*" de forma a obter *feedback* de como o requisito é pretendido pelo cliente e como deve realmente ser implementado.

Pode ser satisfeita definindo, no *Backlog Grooming*, o método de validação de requisitos, e como consequência a criação das *User Stories* para os protótipos, simulações e demonstrações. São usadas iterações curtas e técnicas de *Test-driven Development*.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Realização frequente do *Backlog Grooming* com a participação da equipa.
- Garantir que as User Stories para validação estão documentadas.
- Casos de teste concluídos.

### 5.2.2 Gestão de Requisitos

Em seguida é detalhado, para cada uma das práticas específicas da área de processo *Gestão de Requisitos*, como pode fortalecer o ágil, como o ágil pode satisfazer a prática específica e também como a sua aplicação pode ser demonstrada.

#### 5.2.2.1 SP 1.1 – Entender os requisitos

Esta prática do CMMI permite garantir que os requisitos do negócio e as funcionalidades necessárias são definidas com clareza suficiente para permitir a criação de *Epics* e de *User Stories* isto porque que são fornecidos os critérios de avaliação e aceitação de requisitos mais críticos. É possível garantir o êxito na priorização e implementação das *User Stories*.

De forma a satisfazer esta prática o PO deve desenvolver as *Epics* fornecendo clareza à equipa. O *Backlog Grooming* deve ser realizado com toda a equipa presente onde serão feitas perguntas esclarecedoras de forma levar à compreensão do tamanho e complexidade das *Epics* e das *User Stories* (caso já existam) e também para possibilitar a correta definição dos critérios de aceitação para as *User Stories* criadas.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Realização frequente do *Backlog Grooming* com a participação da equipa.
- Atribuição de *story points* às *Epics* e *User Stories* criadas.
- Garantir a documentação do *Product Backlog* com a presença de *Epics* e de *User Stories*.
- Refinar as *Epics* em *User Stories* menores durante o *Backlog Grooming*.

#### 5.2.2.2 SP 1.2 – Obter compromisso com os requisitos

A identificação de *stakeholders* e a monitorização da sua participação no compromisso com os requisitos irá ajudar a evitar mal-entendidos em relação aos requisitos. Ajuda também a garantir que os requisitos do negócio são claramente entendidos para apoiar a criação de *User Stories*.

Para satisfazer esta prática deve ser realizado o *Backlog Grooming* com a participação de toda a equipa. Desta forma todas as *Epics* e *User Stories* criadas, modificadas ou excluídas são verificadas por todos. No caso das *Epics* permite também a realização de uma discussão até que o seu conteúdo seja claramente compreendido por todos.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Realização frequente do *Backlog Grooming* com a participação da equipa.
- Atribuição de *story points* às *Epics* e *User Stories* criadas.
- Garantir a documentação do *Product Backlog* com a presença de *Epics* e de *User Stories*.
- Refinar as *Epics* em *User Stories* menores durante o *Backlog Grooming*.

#### 5.2.2.3 SP 1.3 – Gerir as alterações nos requisitos

A aplicação desta prática ajuda a reduzir o caos em projetos em que as expectativas do negócio são constantemente ajustadas, garantindo que o valor comercial mais importante é entregue.

Para satisfazer esta prática as *Epics* são modificadas, adicionadas e excluídas do *Product Backlog* pelo PO, sabendo que todas as alterações feitas são discutidas com a equipa.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Realização frequente do *Backlog Grooming* com a participação da equipa.
- Garantir a documentação do *Product Backlog* com a presença de *Epics* e de *User Stories*.

#### 5.2.2.4 SP 1.4 – Manter a rastreabilidade bidirecional dos requisitos

A rastreabilidade bidirecional ajuda a garantir que as *Epics* e *User Stories* já validadas sejam totalmente implementadas, que identifiquem as lacunas existentes e que as *Epics* sejam tratadas de forma adequada antes da implementação. O grande objetivo é enfatizar o valor de entender como as *Epics* são implementadas por meio de *User Stories*.

De forma a satisfazer esta prática deve existir uma rastreabilidade clara entre *Epics* e *User Stories*. Deve ser possível as *User Stories* serem rastreadas para *User Stories* de nível superior (se for o caso) ou para *Epics* de forma a garantir ao PO que as necessidades definidas nas *Epics* são atendidas.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Realização frequente do *Backlog Grooming* com a participação da equipa.
- Mapeamento entre *Epics* e *User Stories*.
- Garantir a documentação do *Product Backlog* com a presença de *Epics* e de *User Stories*.
- Refinar *Epics* em *User Stories* menores nas sessões de *Backlog Grooming* e validar as que já foram refinadas.

#### 5.2.2.5 SP 1.5 – Garantir o alinhamento entre o trabalho e os requisitos

O CMMI relembra a rever constantemente os compromissos, as práticas em uso e como elas suportam a implementação das *User Stories*. Esta prática alimenta também a consideração da utilização de retrospetiva.

Para satisfazer esta prática deve ser realizado o *Backlog Grooming* com todos os elementos da equipa de forma a esclarecer as *User Stories* existentes e se as expectativas relacionadas ao requisito definem as habilidades necessárias à equipa, os ambientes necessários, as arquiteturas necessárias, entre outros.

A correta utilização desta prática pode ser demostrada da seguinte forma:

- Realização frequente do *Backlog Grooming* com a participação da equipa.
- Atribuição de *story points* às *Epics* e *User Stories* criadas.

- Garantir a documentação do *Product Backlog* com a presença de *Epics* e de *User Stories*.

## 5.3 Conclusões

Ao realizar o mapeamento das práticas específicas das áreas de processo *Desenvolvimento de Requisitos* e de *Gestão de Requisitos* do CMMI com metodologias ágeis, foi possível perceber o que o CMMI recomenda para fortalecer o ágil e como aplicar e demonstrar num ambiente ágil.

Para satisfazer o conjunto de práticas específicas das duas áreas de processo estudadas é imprescindível a realização da cerimónia *Backlog Grooming* regularmente com a presença de toda a equipa. Durante a realização da cerimónia será possível para a equipa:

- Identificar novos requisitos e alterações aos existentes.
- Garantir que a equipa fica esclarecida em relação aos requisitos em questões de conceito, dimensão e complexidade.
- Refinar as *Epics* em *User Stories* e proceder à sua documentação no *Product Backlog*.
- Analisar os requisitos existentes e proceder à sua atualização consoante o resultado da análise.
- Atribuir *story points* às *User Stories* especificadas.
- Manter o *Product Backlog* priorizado do mais prioritário para o menos prioritário e respeitando as necessidades do negócio.
- Definir casos de teste e *Definition of Done* para as *User Stories*.

Para tal ser possível, o PO deve fornecer as *Epics* iniciais de forma a garantir a existência de informação que a equipa consiga refinar em objetos de trabalho durante o *Backlog Grooming* e, consequentemente, ser possível criar um *Product Backlog* com *Epics* e *User Stories*. O *Product Backlog* deve ser criado numa ferramenta de gestão ou num quadro físico preparado para esse propósito.

Além do *Backlog Grooming* é importante manter a rastreabilidade bidirecional entre as *Epics* e as *User Stories*. A grande maioria das ferramentas de gestão de projeto já preveem este fator. É importante saber qual a origem dos requisitos, para tal todas as *User Stories* devem manter o registo da *Epic* de onde forma refinadas, criando um mapeamento entre *Epics* e *User Stories* em que seja possível consultar visualmente da hierarquia e relação entre as mesmas. Além disso é possível validar facilmente se as *User Stories* estão a atender as necessidades das *Epics*.

Por fim, é importante garantir o máximo de clareza nas *User Stories* tanto em questões técnicas como do negócio. Cada *User Story* deve conter uma descrição que permite articular o cenário de negócio com o que é realmente para implementar. A descrição deve ainda ser complementada com uma listagem de critérios de aceitação e atributos de qualidade. Com este conjunto de informação, nas *Sprint Rivews* será possível validar se as necessidades e os critérios estão a ser atendidos como planeado.

# 6 Estruturação do Processo

Este capítulo tem como objetivo apresentar a estruturação feita ao processo com base nos problemas encontrados no capítulo 4 e utilizando o mapeamento feito no capítulo 5. Além disso, definir as métricas necessárias para avaliação da estruturação feita ao processo. Na secção 6.1 é apresentado o processo de requisitos proposto neste trabalho sendo mostrado o seu fluxo e como deve ser utilizado pelas equipas. Na secção 6.2 são definidos os objetivos e métricas que irão permitir avaliar a eficiência das alterações ao processo. Em seguida na secção 6.3 é apresentado o mapeamento entre as práticas específicas das áreas de processo do CMMI e as métricas desenvolvidas para a avaliação da eficiência das alterações de forma a verificar a cobertura entre ambas. Por fim na secção 6.4 são apresentadas as conclusões obtidas no presente capítulo.

## 6.1 Processo de Requisitos

Tendo em conta a análise feita anteriormente e comparando com a análise realizada ao processo atual, é possível concluir que existem algumas lacunas que necessitam de ser corrigidas e alguns aspetos melhorados, tais como:

- A especificação dos requisitos (*Epics* e *User Stories*) não é feita corretamente por parte das equipas, o que torna os requisitos pouco claros. Tanto as *Epics* como as *User Stories*, muito raramente, têm mais do que um título/designação.
- A rastreabilidade entre requisitos não é considerada pelas equipas, muito raramente é criada uma relação entre *Epics* e *User Stories*.
- Não existe documentação dos processos nem material de apoio (tais como *templates* ou exemplos reais) tanto para consulta como para treino.
- O *Backlog Grooming* é realizado raramente e quando é realizado não são cumpridas todas as expectativas.
- Ser definido e documentado o processo pelo qual as necessidades do cliente têm de ser sujeitas até chegar ao formato de *Epic* ou de *User Story*.

A informação anteriormente recolhida será usada como guia para o processo de melhoria, existindo o objetivo de assegurar o cumprimento do máximo de práticas do CMMI possíveis tendo em conta o que a organização consegue suportar atualmente e também a capacidade de absorção por parte dos colaboradores. O segundo fator tem um impacto especial porque, atualmente, o nível de utilização de processos, específicos ou gerais, no dia-a-dia por parte dos colaboradores é reduzido.

Durante a construção do processo de requisitos foi necessário considerar vários fatores. Existe a necessidade de trabalhar em conjunto com o Scrum, que é a metodologia adotada, além disso não pode ser muito pesado para os colaboradores e deve ser facilmente compreendido. Em cada atividade existem

responsáveis, apesar disso as atividades podem ser realizadas em conjunto com os restantes intervenientes do projeto.

No Processo inicial, não existe um processo definido nem documentado para a transformação das necessidades do cliente em requisitos com clareza suficiente para alocar ao desenvolvimento, sendo a especificação feita de forma *ad hoc* por parte das equipas. Neste trabalho foi definido um processo que permite transformar as necessidades do cliente e/ou funcionalidades em requisitos no formato de *Epics* e *User Stories*. Esse processo é apresentado na Figura 4 em formato de diagrama com a sequência de atividades que cada funcionalidade ou necessidade do cliente deve percorrer até chegar ao *Product Backlog*. Os intervenientes neste processo são o PO e a ST. Em cada atividade é apresentado o ou os responsáveis, como já referido não implica que a atividade seja executada apenas por esse papel, mas será esse o responsável pelo seu sucesso.

Aplicando este novo processo as equipas irão estar a respeitar a grande maioria das práticas específicas das áreas de processo do CMMI (a cobertura das práticas específicas pode ser consultada na secção 6.3). Nesta fase da otimização não será implementada a definição de casos de teste e *Definition of Done* nas *User Stories*.

Em seguida são detalhadas cada uma das atividades do processo de requisitos estruturado neste trabalho. Para cada uma delas é explicado como deve ser realizada a sua execução e também quando é possível avançar para a atividade seguinte. Todas as atividades apresentadas em seguida correspondem ao processo de requisitos elaborado neste trabalho e apresentado na Figura 4.

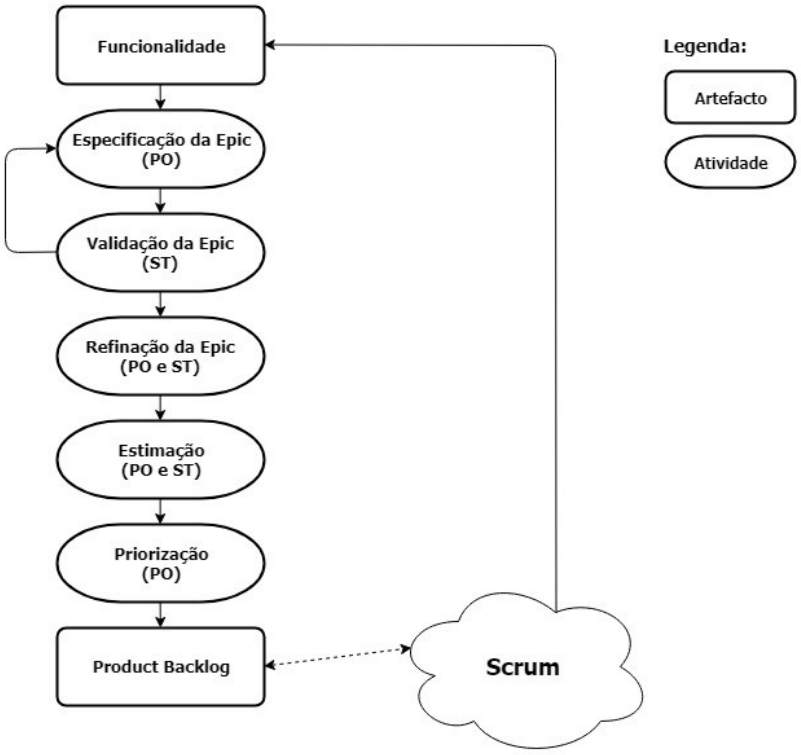



Figura 4 - Processo de requisitos

### 6.1.1 Funcionalidade

Todos os requisitos devem passar por todo o processo de requisitos independentemente da sua origem. Os requisitos podem ter origem de várias fontes:

- Novas necessidades do cliente ou dos *stakeholders*.
- Resultados do processo de elicitação.
- Alterações a requisitos existentes no *Product Backlog*.

Ao considerar novas necessidades do cliente ou dos *stakeholders* e sendo este processo baseado no Scrum, onde são realizadas iterações curtas de forma a rever o trabalho realizado e obter *feedback* em relação ao mesmo frequentemente, é possível satisfazer parte da *SP 1.5 – Validar requisitos* da área de processo de *Desenvolvimento de Requisitos*. Em parte, porque neste trabalho não são previstos os casos de teste para cada *User Story*. Além disso ao considerar as alterações a requisitos existentes no *Product Backlog* está a ser satisfeita a *SP 1.3 – Gerir alterações dos requisitos* da área de processo *Gestão de Requisitos*.

O PO tem a responsabilidade de garantir que os requisitos são colocados o mais rápido possível no processo de forma a que os mesmos cheguem ao *Product Backlog* para serem implementados. Apenas o PO tem a responsabilidade de aceitar novos requisitos ou alterações aos existentes, isto porque é o PO que valida se o requisito encaixa nos objetivos e visão do projeto e também na calendarização negociada com o cliente. Caso a complexidade não permita tomar uma decisão imediata ao PO, pode consultar os *stakeholders* do projeto. Se o requisito não for válido para o desenvolvimento atual, não deve avançar para a especificação.

O comportamento descrito aplica-se tanto ao levantamento inicial de requisitos no procedimento "Pre-game" do Anexo B, como posteriormente no *Backlog Grooming* durante do procedimento "Desenvolvimento" também do Anexo B.

### 6.1.2 Especificação da *Epic*

O principal objetivo desta atividade é melhorar a qualidade do requisito, para tal deve ser respeitada a formatação prevista nos processos da empresa, tornando-o o mais semelhante possível aos restantes e já existentes no *Product Backlog*. O PO deve ter a capacidade de executar a atividade de especificação de *Epics*. Isto porque todos os requisitos criados nesta atividade partilham o mesmo nível de abstração, todos são *Epics*. Posteriormente a ST irá lidar com eles e refinar o quanto acharem necessário.

Sabendo que os requisitos podem ter origem de fontes distintas, o seu tratamento deve ser adequado a cada situação. No caso de ser uma nova necessidade dos *stakeholders* será criada uma nova especificação, no caso de ser uma alteração de uma especificação existente deve apenas ser alterado o seu conteúdo. Para tal pode ser necessário utilizar técnicas de elicitação de requisitos, fazendo com que surjam novas restrições a considerar no desenvolvimento. Caso se verifique essa questão devem ser consideradas na especificação existente, ou se se justificar numa nova especificação.

Para proceder à especificação o PO deve usar a Tabela 2 como guia, respeitando os atributos existentes. É possível consultar a informação completa no Anexo C, tendo sido o mesmo criado no âmbito deste trabalho de forma a dar suporte ao processo de requisitos e também ao PDS no geral.

Tabela 2 - Atributos da *Epic*

| *#* | ***Título*** | ***Descrição*** |
|---|---|---|
| ***1*** | **ID** | Identificador único para a *Epic*. É inserido automaticamente pelo sistema. |
| ***2*** | **Título** | Cada *Epic* deve ter um título que, em resumo, explica a *Epic*. O título não deve ter mais do que uma frase. |
| ***3*** | **Descrição** | A descrição contém informações mais detalhadas sobre a *Epic*. Além disso deve ser descrito o porquê e o benefício da sua implementação no sistema. |
| ***4*** | **Origem** | De onde surgiu a *Epic* Pode ser de uma pessoa, de uma reunião ou de um documento. |
| ***5*** | **Lista de *User Stories* associadas** | Cada *Epic* deve conter a listagem das *User Stories* refinadas através da *Epic* em questão, isto de forma a serem apresentadas todas as suas relações, ajudando a manter a rastreabilidade entre os requisitos. |
| ***6*** | **Criado por** | O elemento que procedeu à definição e criação da *Epic* no *Product Backlog*. É inserido automaticamente pelo sistema. |
| *7* | **Data de criação** | Data de criação. É inserido automaticamente pelo sistema. |
| ***8*** | **Data de atualização** | Data da última atualização. É inserido automaticamente pelo sistema. |

Dos atributos enumerados na Tabela 2, o "ID", o "Criado por", a "Data de criação" e a "Data de atualização" são criados e geridos automaticamente pela ferramenta de gestão de projeto. O atributo "Lista de *User Stories* associadas" é preenchido à medida que as *User Stories* vão sendo refinadas e associadas à *Epic* respetiva. Por sua vez faltam o "Título", a "Descrição" e a "Origem" para o PO preencher de forma a completar a especificação da *Epic*.

O atributo "Descrição" deve ser o mais completo e claro que seja possível no momento da especificação, isto porque todo o processo de refinação vai girar em torno dessa descrição. O atributo "Origem" mesmo sendo opcional deve ser preenchido sempre que possível, de forma a garantir que não existirão dúvidas de quem, quando e em que circunstancia surgiu o requisito, sendo possível no futuro perceber o porquê da existência de cada *Epic*.

Nesta atividade é possível satisfazer a prática específica *SP 1.1 – Elicitação de necessidades* da área de processo *Desenvolvimento de Requisitos* já que o PO apresenta e especifica as *Epics* a serem refinadas posteriormente em conjunto com a ST.

### 6.1.3 Validação da *Epic*

Antes de avançar, a ST deve proceder à validação da especificação da *Epic*. Esta validação tem como objetivo garantir que existe informação suficiente para proceder ao processo de refinação e em seguida produzir uma implementação útil para o cliente. Não é necessário a *Epic* estar totalmente completa, mas deve apresentar o mínimo de clareza para a ST e o PO a conseguir refinar ao ponto de, pelo menos,

conseguir criar um protótipo daquilo que é esperado pelo cliente. Isto sabendo que no futuro poderá sofrer alterações ou ser completada com mais informação.

Caso a ST não considere que existe o mínimo de clareza na especificação da *Epic* (diferentes projetos podem ter diferentes níveis de clareza), mesmo para uma fase inicial, deve ser rejeitada e o PO deve procurar complementar a especificação. Mesmo que para isso necessite de recorrer novamente a técnicas de elicitação de requisitos. Quando esta atividade esta a ser realizada no procedimento "Pre-game" do Anexo B pode ser comum existirem várias *Epics* rejeitadas pela ST.

Nesta atividade é possível satisfazer a *SP 3.3 – Analisar requisitos* da área de processo *Desenvolvimento de Requisitos* uma vez que os requisitos são validados pela ST e sujeitos a atualizações consoante a análise realizada. A *SP 1.2 – Obter compromisso com os requisitos* da área de processo *Gestão de Requisitos* também é satisfeita uma vez que ao validar um requisito a ST esta a assumir que o mesmo é claro o suficiente para ser refinado.

### 6.1.4 Refinação da *Epic*

Após da *Epic* ser validada a ST em conjunto com o PO podem proceder à sua refinação em *User Stories* menores. Esta atividade não prevê que sejam criadas todas as *User Stories* possíveis, esse processo pode ser realizado iterativamente durante as cerimónias de *Backlog Grooming*. Deve ser especificado o que a ST e o PO acharem aceitável e possível naquela fase, garantindo que as *User Stories* são claras o suficiente para serem implementadas. Mais tarde, caso seja necessária mais informação, a *Epic* irá ser colocada novamente no início deste processo, de forma ao PO recorrer as técnicas de elicitação previstas no PDS (Anexo B) e assim garantir a clareza necessária.

Para proceder à especificação a ST e o PO devem usar a Tabela 3 como guia, respeitando os atributos existentes. É possível consultar a informação completa no Anexo D, tendo sido o mesmo criado no âmbito deste trabalho de forma a dar suporte ao processo de requisitos e também ao PDS no geral.

Tabela 3 - Atributos da *User Story*

| *#* | ***Título*** | ***Descrição*** |
|---|---|---|
| ***1*** | **ID** | Identificador único para a *User Story*. É inserido automaticamente pelo sistema. |
| ***2*** | **Título** | Cada *User Story* deve ter um título que, em resumo, explica a *User Story*. O título não deve ter mais do que uma frase. |
| ***3*** | **Descrição** | A descrição contém informações mais detalhadas sobre a *User Story*. Além disso deve ser descrito o porquê e o benefício da sua implementação no sistema na ótica do negócio. |
| ***4*** | **Critérios de aceitação** | Cada *User Story* deve conter um conjunto de critérios de aceitação que permitem à equipa perceber todos os critérios que devem ser cumpridos na implementação da *User Story*. |
| ***5*** | **Relação** | Deve existir uma relação com a *Epic* de onde teve origem o processo de refinação. |
| ***6*** | **Lista de tarefas** | Quando a equipa procede ao desenvolvimento da *User Story* deve criar uma lista de tarefas com todos os passos necessários para o desenvolvimento da *User Story*. Acaba por ser um pequeno relatório das atividades realizadas. |
| ***7*** | ***Story Points*** | Número de *Story Points* atribuídos pela equipa no *Backlog Grooming*. Os *Story Points* utilizados devem respeitar a sequência de Fibonacci. |
| ***8*** | **Plataforma** | É representada por uma *tag* que classifica o tipo de plataforma em que o requisito irá ser desenvolvido. |
| ***9*** | **Tempo gasto** | Após a implementação deve ser feito o registo do tempo gasto na implementação da *User Story* em horas. |
| ***10*** | ***Sprint* e estado** | Quando uma *User Story* é alocada a um Sprint deve ser registado qual o Sprint a que foi alocada tal como o estado em que se encontra. |
| ***11*** | **Criado por** | O elemento que procedeu à definição e criação da *User Story* no *Product Backlog*. É inserido automaticamente pelo sistema. |
| ***12*** | **Data de criação** | Data de criação. É inserido automaticamente pelo sistema. |
| ***13*** | **Data de atualização** | Data da última atualização. É inserido automaticamente pelo sistema. |

Dos atributos enumerados na Tabela 3, o "ID", o "Criado por", a "Data de criação" e a "Data de atualização" são criados e geridos automaticamente pela ferramenta de gestão de projeto. O atributo "*Sprint* e estado" é preenchido automaticamente quando a *User Story* é alocada a um Sprint e cada vez que o seu estado é alterado no respetivo *Sprint*. Relativamente ao atributo "Relação", deve ser adicionada a referência da *Epic* da qual foi refinada, garantindo assim a rastreabilidade e também a possibilidade de criar um mapeamento visual da hierarquia dos requisitos existentes no *Product Backlog*. Por sua vez faltam o "Título", a "Descrição", os "Critérios de aceitação" e a "Plataforma" para serem preenchidos nesta atividade.

Os atributos "Descrição" e "Critérios de aceitação" são possivelmente os atributos mais importantes das *User Stories*, pois a sua má definição aumentará a probabilidade de insucesso na implementação. O objetivo do atributo "Descrição" não é apenas que sejam detalhados os aspetos técnicos, devem também articular o cenário de negócio com o que é realmente para implementar. Relativamente aos "Critérios de aceitação" é importante conseguir garantir tudo o que é necessário para conseguir disponibilizar o sistema ao cliente tal como pretendido, desta forma são definidos os critérios de aceitação

e atributos de qualidade que a ST deve ter em conta durante a implementação. Posteriormente irão ser usados como suporte de validação das demonstrações realizadas.

Por fim o atributo "Plataforma" diz respeito a uma *tag* que representa o tipo de plataforma para o qual a *User Story* é direcionada (por exemplo: *back-end* ou *mobile*). A sua definição é opcional, visto que apenas tem como objetivo categorizar visualmente a *User Story* na ferramenta de gestão de projeto. Torna-se mais relevante e útil em projetos que agregam múltiplas plataformas, por exemplo uma aplicação móvel que requeira uma API para questões de autenticação e acesso aos dados.

Nesta atividade é possível satisfazer um grande conjunto de práticas de ambas as áreas de processo *Desenvolvimento de Requisitos* e *Gestão de Requisitos*. Ao proceder à refinação das *Epics* em *User Stories* menores garantindo o mapeamento e rastreabilidade podem ser satisfeitas as práticas *SP 1.2 – Transformar necessidades dos stakeholders* em requisitos do cliente e *SP 1.4 – Manter a rastreabilidade bidirecional dos requisitos*. O processo de requisitos pode ser iniciado não só pela criação de novas *Epics* como também pela alteração das existentes, isso pode ser refletido na criação de novas *User Stories* satisfazendo a prática *SP 2.1 – Estabelecer os requisitos de produto e componente de produto*. Por fim a estrutura criada para as *User Stories* na Tabela 3 permite satisfazer as seguintes práticas:

- *SP 2.2 – Alocar requisitos dos componentes de produto* devido à existência de uma lista de tarefas.
- *SP 3.1 – Estabelecer conceitos e cenários operacionais* devido à existência de uma descrição do cenário de negócio.
- *SP 3.3 – Estabelecer definições de funcionamento necessário e atributos de qualidade* devido à existência de critérios de aceitação.
- *SP 1.1 – Entender os requisitos* devido à existência de critérios de aceitação e de *story points*.

### 6.1.5 Estimação

Nesta atividade é feita a estimativa das *User Stories*. Como previsto no Anexo B as estimativas são feitas em *story points*, e tanto o PO como a ST participam nesta atividade. O objetivo é preencher o atributo "*Story Points*" da Tabela 3, para tal é usada a técnica p*lanning poker*. O preenchimento só acontece e se os intervenientes chegarem a uma pontuação unânime, ou no pior dos casos maioritária caso não seja possível chegar a um consenso. Por fim esta atividade não deve terminar enquanto existirem *User Stories* no *Product Backlog* por estimar.

Nesta atividade é possível satisfazer as práticas *SP 1.1 – Entender os requisitos* e *SP 1.5 – Garantir o alinhamento entre o trabalho e os requisitos* da área de processo *Gestão de Requisitos* uma vez que é feita a atribuição de *story points* às *User Stories*.

### 6.1.6 Priorização

O *Product Backlog* deve ter sempre o seu conteúdo ordenado do mais prioritário para o menos prioritário. Respeitando esta prática os requisitos mais prioritários para o negócio estarão sempre no topo, mas nunca no mesmo nível. Isto é, por cada nível só existe uma *User Story*, não existem duas *User Stories* exatamente com a mesma prioridade.

Esta atividade é da responsabilidade do PO, sabendo que quando procede à priorização do *Product Backlog* deve ter em consideração o que irá trazer mais valor ao negócio.

Nesta atividade é possível satisfazer a prática *SP 3.4 – Analisar os requisitos para alcançar o equilíbrio* da área de processo *Desenvolvimento de Requisitos* uma vez que será garantido que o *Product Backlog* irá ter as *User Stories* mais prioritárias no topo.

### *6.1.7 Product Backlog*

Por fim todas as *Epics* e *User Stories* acabam por ser armazenadas no *Product Backlog*. Caso existam alterações que influenciem as *Epics* e *User Stories* existentes no *Product Backlog*, as *Epics* afetadas devem ser submetidas novamente ao processo, percorrendo todas as atividades. As alterações realizadas nas *Epics* poderão ter influência nas *User Stories* existentes no *Product Backlog* e que ainda não foram implementadas. Nesses casos o PO e ST devem proceder em conformidade e avançar para a modificação ou exclusão das mesmas.

No *Sprint Planning*, quando uma *User Story* é alocada a um Sprint a equipa deve proceder ao preenchimento do atributo "Lista de tarefas" da Tabela 3. Já durante o Sprint o tempo gasto por cada elemento da ST na implementação e na realização de testes às *User Stories* deve ser registado no atributo "Tempo gasto". Esse registo deve ser feito em horas na ferramenta de gestão, que permite acumular os tempos introduzidos.

Por fim, o fator de conclusão de uma *User Story* é representado pelo atributo "*Sprint* e estado" quando este atributo contém o *Sprint* onde foi realizada a implementação da *User Story* concatenado com o estado *Done*. Por sua vez o fator de conclusão de uma *Epic* é verificado através das *User Stories* a si associadas, sendo dada como concluída quando todas elas estiverem concluídas e abrangerem toda a funcionalidade prevista até ao momento.

## 6.2 Definição de Métricas para Avaliação

Para validar se as alterações ao processo se estão a refletir em otimizações ou não, foram definidas um conjunto de métricas que permitem avaliar a eficiência das alterações realizadas. A abordagem utilizada é o GQM que fornece uma técnica de medição intencional.

Na primeira fase foi desenvolvido o nível conceptual que diz respeito aos objetivos. Cada objetivo representa um objeto de medição e são definidos para medir a eficiência das alterações feitas ao processo. Após a definição dos objetivos foi desenvolvido o nível operacional que diz respeito as questões e que pretendem caracterizar os objetos de medição em relação ao problema existente. Foi também desenvolvido

o nível quantitativo que pretende responder às questões elaboradas. Com base nas áreas de processo *Desenvolvimento de Requisitos* e *Gestão de Requisitos* do CMMI, foi definido o seguinte:

Objetivo 1. **Aproximar os requisitos das necessidades do cliente.**

Q1. *Com que regularidade é recolhido feedback do cliente?*

M1. Regularidade com que é recolhido feedback do cliente.

Q2. *São definidas novas Epics após o início do projeto?*

M2. Percentagem de *Epics* definidas após o início do projeto.

Objetivo 2. **Melhorar o processo de transformação das necessidades do cliente em requisitos.**

Q3. *As Epics são definidas no início do projeto?*

M3. Percentagem de *Epics* definidas no início do projeto.

Q4. *As Epics são divididas em User Stories menores?*

M4. Percentagem de *Epics* que foram refinadas em *User Stories menores*.

Objetivo 3. **Melhorar a forma como os requisitos são especificados e documentados na ferramenta de gestão de projeto.**

Q5. *Com que regularidade é feita a cerimónia Backlog Grooming?*

M5. Regularidade com que é realizada a cerimónia de *Backlog Grooming*.

Q6. *As User Stories apresentam a descrição do cenário de negócio?*

M6. Percentagem de *User Stories* que apresentam a descrição do cenário de negócio.

Q7. *As User Stories apresentam a identificação do tipo de plataforma para o qual a User Story é direcionada (exemplo: Back-end, Front-end, Mobile, etc.)?*

M7. Percentagem de *User Stories* que apresentam a identificação do tipo de plataforma.

Q8. *A definição, alteração, exclusão e priorização de Epics e User Stories do Product Backlog ocorre durante o Backlog Grooming?*

M8. Regularidade com que a definição, alteração, exclusão e priorização de *Epics* e *User Stories* é realizada durante o *Backlog Grooming*.

Q9. *Todas as User Stories e Epics a ser implementadas são documentados na ferramenta de gestão e validadas durante Backlog Grooming?*

M9. Regularidade com que a documentação e validação das *Epics* e *User Stories* acontece durante *Backlog Grooming*.

Objetivo 4. **Reduzir o número de inconsistências ente os requisitos definidos e a sua respetiva implementação.**

Q10. *As Epics apresentam a descrição do que é pretendido de acordo com as necessidades do cliente?*

M10. Percentagem de *Epics* que apresentam descrição.

Q11. *As User Stories são definidas com clareza suficiente?*

M11. Clareza com que são definidas as *User Stories*.

Objetivo 5. **Garantir que todas as User Stories oferecem suporte às Epics definidas**.

Q12. *Todas as User Stories são refinadas através de Epics?*

M12. Percentagem de *User Stories* que apresentam a relação com uma *Epic*.

Objetivo 6. **Melhorar a forma como os critérios de conclusão dos requisitos são definidos.**

Q13. *As User Stories apresentam a descrição dos critérios de aceitação?*

M13. Percentagem de *User Stories* que apresentam a descrição dos critérios de aceitação.

Q14. *As Users Stories apresentam uma lista de tarefas que definem a sua funcionalidade?*

M14. Percentagem de *User Stories* que apresentam uma lista de tarefas.

Objetivo 7. **Melhorar a forma como é feita a gestão de alterações das Epics e User Stories.**

Q15. *As Epics são definidas, alteradas e excluídas pelo Product Owner sendo cada alteração discutida com a equipa?*

M8. Regularidade com que a definição, alteração, exclusão e priorização de *Epics* e *User Stories* é realizada durante o *Backlog Grooming*.

Q16. *Com que regularidade as alterações nas Epics têm influência em outras Epics já definidas?*

M15. Regularidade com que as alterações numa *Epic* influenciam a definição de outras *Epics*.

Q17. *Quais são os motivos que provocam alterações nas Epics?*

M16. Motivos que provocam alterações nas *Epics*.

Objetivo 8. **Manter a rastreabilidade dos requisitos ao longo do desenvolvimento do projeto.**

Q18. *As User Stories definidas têm referência para a Epic através da qual foram definidas?*

M12. Percentagem de *User Stories* que apresentam a relação com uma *Epic*.

Objetivo 9. **Melhorar a forma como é feita a gestão dos tempos de execução das Epics e das User Stories.**

Q19. *As User Stories apresentam a descrição do tempo gasto, em horas, na sua implementação?*

M17. Percentagem de *User Stories* que apresentam a descrição do tempo gasto na implementação.

Q20. *Qual o tempo médio em horas gasto para implementar uma User Story?*

M18. Tempo médio em horas para implementar uma *User Story*.

Q21. *Qual o tempo médio em horas gasto para implementar uma Epic?*

M19. Tempo médio em horas para implementar uma *Epic*.

Objetivo 10: **Medir o quão bem a equipa se concentra nas necessidades do projeto.**

Q22. *Com que regularidade a priorização do Product Backlog é afetada devido à não disponibilização de ferramentas ou informações necessárias ao desenvolvimento?*

M20. Regularidade com que a priorização do *Product Backlog* é afetada devido a fatores externos ao desenvolvimento.

Será com base neste conjunto de objetivos, questões e métricas que a eficiência das alterações feitas ao processo será medida.

## 6.3 Mapeamento das áreas de processo com as métricas desenvolvidas

Não será possível satisfazer todas as práticas específicas das duas áreas de processo do CMMI estudas, apesar disso a grande maioria irá ser satisfeita. A verificação da sua utilização pode ser feita consultado a plataforma de gestão ou então consultando os resultados obtidos em cada uma das métricas tendo em conta o seguinte mapeamento:

Tabela 4 - Mapeamento de práticas especificas do CMMI para métricas de QGM

| *Área de Processo* | *Prática específica do CMMI* | *Métrica GQM* |
|---|---|---|
| ***Desenvolvimento de Requisitos*** | SP 1.1 – Elicitação de necessidades | M3; M4; M5; M1 |
| | SP 1.2 – Transformar as necessidades dos *stakeholders* em requisitos do cliente | M4; M5; M8 |
| | SP 2.1 – Estabelecer os requisitos de produto e de componentes de produto | M8 |
| | SP 2.2 – Alocar requisitos dos componentes de produto | M14 |
| | SP 2.3 – Identificar os requisitos de interface | - |
| | SP 3.1 – Estabelecer conceitos e cenários operacionais | M6; M8 |
| | SP 3.2 – Estabelecer definições de funcionamento necessário e atributos de qualidade | M13; M8 |
| | SP 3.3 – Analisar requisitos | M10; M9 |
| | SP 3.4 – Analisar os requisitos para alcançar o equilíbrio | M8 |
| | SP 3.5 – Validar requisitos | M5; M1 |
| ***Gestão de Requisitos*** | SP 1.1 – Entender os requisitos | M8; M9; M11 |
| | SP 1.2 – Obter compromisso com os requisitos | M3; M5; M9 |
| | SP 1.3 – Gerir as alterações nos requisitos | M8; M2; M15; M16 |
| | SP 1.4 – Manter a rastreabilidade bidirecional dos requisitos | M4; M12 |
| | SP 1.5 – Garantir o alinhamento entre o trabalho e os requisitos | M5; M11 |

## 6.4 Conclusões

Através da estruturação realizada neste capítulo foi possível otimizar e fortalecer a utilização do Scrum utilizando duas áreas de processo do CMMI, *Desenvolvimento de Requisitos* e *Gestão de Requisitos*. O processo de requisitos definido abrange todas as práticas específicas da *Gestão de Requisitos* e quase todas do *Desenvolvimento de Requisitos*, apenas não contemplando a *SP 2.3 – Identificar requisitos de interface* e alguns tópicos de outras práticas em questões relacionadas com os casos de teste e *Definition of Done*.

No decorrer deste trabalho foi possível perceber a importância do *Backlog Grooming* no desenvolvimento e gestão dos requisitos, tal como da existência de *templates* e materiais de apoio que podem ser consultados para que a especificação seja feita de forma normalizada de equipa para equipa e de projeto para projeto. Desta forma foram desenvolvidos *templates* e exemplos práticos para cada um dos *templates* para que as equipas consigam consultar em caso de dúvida e também para a formação de novos colaboradores.

A definição das métricas para validar se as alterações ao processo se refletem em otimizações ou não também permite verificar se as diferentes práticas específicas das áreas de processo *Desenvolvimento de Requisitos* e *Gestão de Requisitos* são satisfeitas ou não. O mapeamento resultante pode ser consultado na Tabela 4.

# 7 Aplicação e Avaliação do Estudo Realizado

Este capítulo tem como objetivo avaliar as alterações realizadas ao processo. Para tal foram selecionados um conjunto de projetos da empresa de forma a obter resultados. O processo de aplicação e avaliação do estudo realizado foi dividido nas seguintes fases que são apresentadas nas seguintes secções. Na secção 7.1 é realizada a preparação da aplicação do estudo e desenvolvimento da *dashboard* de análise. Em seguida, na secção 7.2 foi feita a aplicação do estudo através da realização de um projeto com base no processo estruturado e sugerido neste trabalho. Na secção 7.3, após o término do projeto, foram analisados os resultados e feita uma análise comparativa entre o antes e o após alterações. Por fim na secção 7.4 são apresentadas as principais conclusões da aplicação e da avaliação.

## 7.1 Preparação da Aplicação do Estudo

Quando um projeto termina é extraída toda a informação possível da ferramenta de gestão de projetos de forma a realizar o cálculo das métricas objetivas, além disso são realizados inquéritos obter a informação necessária para o cálculo das métricas objetivas. Os dados para o cálculo das métricas objetivas, consoante a dimensão do projeto, podem ascender a um número que se torna bastante complexo para uma análise manual. Desta forma foi desenvolvido um sistema que permite o carregamento dos dados obtidos através da ferramenta de gestão de projeto e dos inquéritos realizados. Após a recolha é possível consultar um *dashboard* com todas as métricas calculadas, a média dos projetos inseridos até ao momento e o objetivo definido pela gestão para cada métrica. Os resultados dos inquéritos também são apresentados nesse mesmo *dashboard*.

Estando definidas um conjunto de métricas na secção 6.2 para a avaliação das alterações feitas ao processo neste trabalho, é necessário perceber como é que serão obtidos valores para cada uma delas. Existem dois grupos: as métricas calculadas através dos dados fornecidos pela ferramenta de gestão de projetos; e as métricas calculadas através dos inquéritos realizados aos colaboradores após o término do projeto.

Foram isoladas as métricas subjetivas na preparação dos inquéritos. Isto, de forma a possibilitar uma resposta o mais real possível foi necessário categorizar os valores que cada uma delas pode assumir, sendo o resultado apresentado em seguida:

- M1. Regularidade com que é recolhido *feedback* do cliente.
    1. Nunca é recolhido *feedback*.
    2. Apenas no final quando é feita a primeira entrega do projeto.
    3. Raramente no final de cada *Sprint* (em menos de 50% dos *Sprints*).

4. Regularmente no final de cada *Sprint* (em mais de 50% dos *Sprints*, mas não na totalidade).
5. Sempre no final de cada *Sprint* (100% dos *Sprints*)

- M5. Regularidade com que é realizada a cerimónia de *Backlog Grooming*.
  1. Nunca é realizado (em 0% dos *Sprints*).
  2. Raramente no final de cada *Sprint* (em menos de 50% dos *Sprints*).
  3. Regularmente no final de cada *Sprint* (em mais de 50% dos *Sprints*, mas não na totalidade).
  4. Sempre no final de cada *Sprint* (em 100% dos *Sprints*).
  5. A qualquer altura sempre que existe necessidade da sua realização.
- M8. Regularidade com que a definição, alteração, exclusão e priorização de *Epics* e *User Stories* é realizada durante o *Backlog Grooming*.
  1. Nunca são realizadas durante o *Backlog Grooming*.
  2. São realizadas durante o *Backlog Grooming* apenas uma vez no início do projeto.
  3. São realizadas durante o *Backlog Grooming* apenas após o início do projeto.
  4. São realizadas sempre durante o *Backlog Grooming*.
- M9. Regularidade com que a documentação e validação das *Epics* e *User Stories* acontece durante *Backlog Grooming*.
  1. Nunca são realizadas durante o *Backlog Grooming*.
  2. São realizadas durante o *Backlog Grooming* apenas uma vez no início do projeto.
  3. São realizadas durante o *Backlog Grooming* apenas após o início do projeto.
  4. São realizadas sempre durante o *Backlog Grooming*.
- M15. Regularidade com que as alterações numa *Epic* influenciam a definição de outras *Epics*.
  1. Nunca (0% das vezes que existem alterações).
  2. Raramente (até 25% das vezes que existem alterações).
  3. Às vezes (até 50% das vezes que existem alterações).
  4. Muitas vezes (até 75% das vezes que existem alterações).
  5. Sempre (100% das vezes que existem alterações).

- M16. Motivos para alteração nas *Epics*.
    1. Elicitação realizada inicialmente foi incompleta ou insuficiente.
    2. A definição da *Epic* não foi clara o suficiente.
    3. Os requisitos foram mal percebidos pela equipa.
    4. Surgiram necessidades complementares no âmbito do projeto.
    5. Surgiram dificuldades técnicas não previstas pela equipa.
    6. Outro
- M20. Regularidade com que a priorização do *Product Backlog* é afetada devido a fatores externos ao desenvolvimento.
    1. Nunca (nunca é afetada por fatores externos).
    2. Raramente (até 25% das vezes que existem fatores externos).
    3. Às vezes (até 50% das vezes que existem fatores externos).
    4. Muitas vezes (até 75% das vezes que existem fatores externos).
    5. Sempre (sempre que existem fatores externos).

Relativamente às métricas objetivas serão calculadas através dos dados fornecidos pela ferramenta da gestão de projeto. A informação é exportada em quatro ficheiros, cada um deles armazenando diferentes informações sobre os requisitos. O primeiro armazena toda a informação relativa às *Epics*, o segundo relativa às *User Stories*, o terceiro relativo às tarefas realizadas na implementação das *User Stories* e por fim o quarto com a informação dos *bugs* reportados.

No seguimento de uma reunião com os colaboradores responsáveis pela gestão de projeto foram definidos objetivos a alcançar em cada métrica nos projetos. Na Tabela 5 são representados esses objetivos.

Tabela 5 - Objetivos definidos para métricas GQM

| ***Código*** | ***Métrica*** | ***Objetivo*** |
|---|---|---|
| ***M1*** | Regularidade com que é recolhido *feedback* do utilizador/cliente: | Sempre no final de cada *Sprint* (100% dos *Sprints*) |
| ***M4*** | Percentagem de *Epics* que foram subdivididas em *User Stories* menores | 100% |
| ***M5*** | Regularidade com que é realizada a cerimónia de *Backlog Grooming* | A qualquer altura sempre que existe necessidade da sua realização |
| ***M6*** | Percentagem de *User Stories* que apresentam a descrição do cenário de negócio | 100% |
| ***M7*** | Percentagem de *User Stories* que apresentam a identificação do tipo de plataforma | 100% |
| ***M8*** | Regularidade com que a definição, alteração, exclusão e priorização de *Epics* e *User Stories* é realizada durante o *Backlog Grooming* | São realizadas sempre durante o *Backlog Grooming* |
| ***M9*** | Regularidade com que a documentação e validação das *Epics* e *User Stories* acontece durante *Backlog Grooming* | São realizadas sempre durante o *Backlog Grooming* |
| ***M10*** | Percentagem de *Epics* que apresentam descrição | 100% |
| ***M11*** | Clareza com que são definidas as *User Stories* | 95% |
| ***M12*** | Percentagem de *User Stories* que apresentam a relação com uma *Epic* | 100% |
| ***M13*** | Percentagem de *User Stories* que apresentam a descrição dos critérios de aceitação | 90% |
| ***M14*** | Percentagem de *User Stories* que apresentam uma lista de tarefas | 100% |
| ***M15*** | Regularidade com que as alterações numa *Epic* influenciam a definição de outras *Epics* | Raramente (até 25% das vezes que existem alterações) |
| ***M17*** | Percentagem de *User Stories* que apresentam a descrição do tempo gasto na implementação | 100% |
| ***M20*** | Regularidade com que a priorização do *Product Backlog* é afetada devido a fatores externos ao desenvolvimento | Raramente (até 25% das vezes que existem fatores externos) |

Por fim e após toda a informação necessária estar organizada, definida e validada, foi implementado um pequeno sistema que realiza os cálculos necessários para cada uma das métricas definidas e apresenta os resultados no *dashboard*. Cada projeto inserido tem o seu próprio *dashboard*, apesar disso, em algumas das métricas é apresentado o valor médio de todos os projetos inseridos até ao momento. Desta forma é possível, para os gestores de projeto, conseguirem verificar o nível de eficiência do projeto e também comparativamente à eficiência geral dos projetos realizados na empresa.

## 7.2 Aplicação do Estudo

Antes de proceder à atualização do processo ao nível organizacional é necessário verificar se as alterações representam melhorias e se consegue resolver os problemas identificados para resolução. Para tal será desenvolvido um projeto de um cliente em que serão usadas as alterações ao processo definidas neste

trabalho. O objetivo será comparar os resultados com os projetos anteriores e verificar se as alterações ao processo cumprem os objetivos definidos e se representam melhorias tanto para a gerência como para os colaboradores. Os seguintes projetos foram realizados e terminados antes deste estudo:

- Projeto 1 – App híbrida com *Voice over Internet Protocol* (VoIP*)*.
- Projeto 2 – Plataforma Web.
- Projeto 3 – App guia turístico.
- Projeto 4 – Loja *online*.

Já o "Projeto 5 – App catálogo digital" foi realizado após a estruturação ao processo tendo sido aplicadas as alterações sugeridas. Em seguida são detalhados cada um deles.

### 7.2.1 Projeto 1 – App híbrida com VoIP

Consiste em três sistemas distintos. Uma App Android e iOS desenvolvida com uma *framework* híbrida, um *backoffice* para gestão de todo o sistema e uma *Representational State Transfer* (REST) *Application Programming Interface* (API) que tanto a App como o *backoffice* integram de forma a conseguirem trabalhar a informação do sistema. Foram usados serviços externos para a realização de VoIP e comunicação por chat entre utilizadores. Este projeto contou com uma equipa de três elementos.

### 7.2.2 Projeto 2 – Plataforma Web

Consiste no desenvolvimento de um *Minimum Viable Product* (MVP) para uma plataforma de gestão de projetos de construção civil. Foram usadas tecnologias Web para cliente e servidor. Este projeto contou com uma equipa de dois elementos.

### 7.2.3 Projeto 3 – App guia turístico

Consiste num guia turístico para um município em que forma desenvolvida três plataformas distintas. Uma App Android e iOS desenvolvida com uma *framework* híbrida, um *backoffice* para gestão de todo o sistema e uma REST API que tanto a App como o *backoffice* integram para conseguirem trabalhar a informação. Este projeto contou com uma equipa de quatro elementos.

### 7.2.4 Projeto 4 – Loja *online*

Consiste numa loja online de roupas. Foram usados serviços externos para lidar com pagamentos e com lógica de funcionamento base de uma loja online. Apenas foi desenvolvida para a Web. Este projeto contou com uma equipa de quatro elementos.

### 7.2.5 Projeto 5 – App catálogo digital

O projeto a ser realizado consiste numa App para Android e iOS desenvolvida com a *framework* híbrida. A App integra uma REST API desenvolvida com o propósito de fornecer conteúdo à App e também para permitir a gestão desse mesmo conteúdo através de um *backoffice* na Web. Este projeto será realizado por uma equipa de dois elementos. As alterações feitas ao processo foram expostas e explicadas apenas aos

intervenientes deste projeto, sendo também fornecida a documentação existente até ao momento, incluindo a configuração da plataforma de gestão de projetos de acordo com a proposta feita neste trabalho.

## 7.3 Avaliação do Estudo Realizado

A avaliação do estudo irá ter como base um conjunto de projetos realizados anteriormente à alteração do processo e um projeto realizado já com as alterações feitas ao processo, mais especificamente na gestão de desenvolvimento de requisitos. A avaliação irá ser suportada pelas métricas já definidas utilizando o GQM. Em seguida são apresentados os resultados, são apresentados de acordo com cada objetivo definido. Em todos os resultados apresentados é feita uma apreciação qualitativa dos resultados.

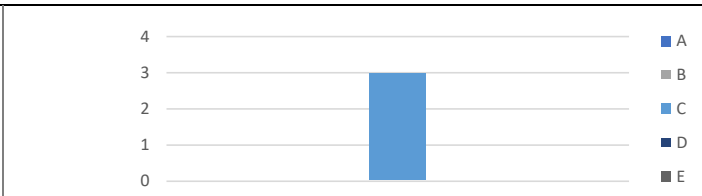
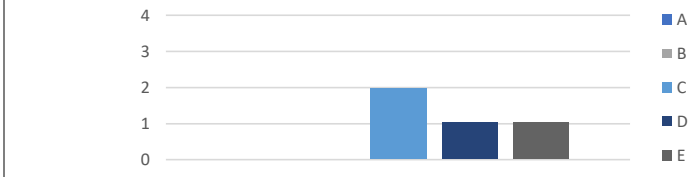
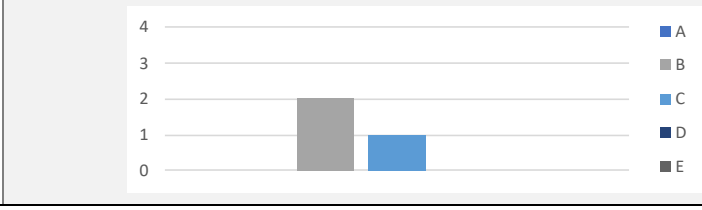
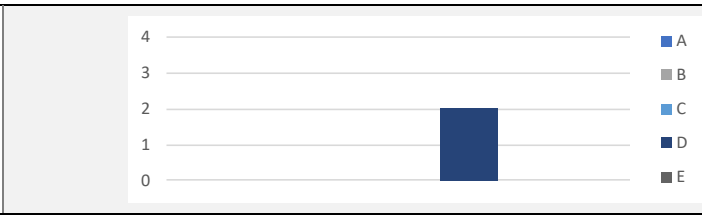
### 7.3.1 Objetivo 1. Aproximar os requisitos das necessidades do cliente

Q1. Com que regularidade é recolhido *feedback* do cliente?

Os gráficos apresentados na seguinte tabela regem-se pelas seguintes opções:

- **A** – Nunca é recolhido *feedback*.
- **B** – Apenas no final quando é feita a primeira entrega do projeto.
- **C** – Raramente no final de cada *Sprint* (em menos de 50% dos *Sprints*).
- **D** – Regularmente no final de cada *Sprint* (em mais de 50% dos *Sprints*, mas não na totalidade).
- **E** – Sempre no final de cada *Sprint* (100% dos *Sprints*)

Tabela 6 - M1. Regularidade com que é recolhido *feedback* do cliente

| ***Projeto*** | ***M1. Regularidade com que é recolhido feedback do cliente*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** |  |
| ***Projeto 2 – Plataforma Web*** |  |
| ***Projeto 3 – App guia turístico*** |  |
| ***Projeto 4 – Loja online*** |  |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** |  |

| | |
|---|---|
| ***Objetivo*** | Sempre no final de cada *Sprint* (100% dos *Sprints*) |

Q2. São definidas novas *Epics* após o início do projeto?

Tabela 7 - M2. Percentagem de *Epics* definidas após o início do projeto

| ***Projeto*** | ***M2. Percentagem de Epics definidas após o início do projeto*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 16.7% |
| ***Projeto 2 – Plataforma Web*** | 0% |
| ***Projeto 3 – App guia turístico*** | 0% |
| ***Projeto 4 – Loja online*** | 8.3% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 8.3% |
| | |
| ***Média*** | 6.7% |

As equipas devem aproximar o máximo possível os requisitos às necessidades do cliente, de forma a garantir o máximo de satisfação quando o sistema a ser desenvolvido for entregue. Para tal é deve ser recolhido *feedback* constantemente com base no trabalho desenvolvido até ao momento, mas para isso é necessário que sejam definidos um conjunto de requisitos no início do projeto de forma a dar um base desenvolvimento à equipa. Foi definido que as equipas devem recolher *feedback* no final de cada Sprint. Assim é possível perceber se as suas necessidades estão a ser atendidas e também receber novas necessidades e prioridades do cliente.

Na Tabela 6 é possível verificar que a prática é realizada, no entanto existe alguma dificuldade por parte das equipas em cumprirem o objetivo definido, mesmo no "Projeto 5 – App catálogo digital" onde já foram aplicadas as alterações ao processo.

Relativamente à definição de *Epcis* após o início do projeto não representa necessariamente algo negativo, mas sim a capacidade da equipa em acompanhar o que o cliente precisa ao longo do processo de desenvolvimento. Os motivos podem ser vários, como por exemplo surgirem novas necessidades ou mesmo a elicitação inicial não ter sido concluída na totalidade devido à falta de informação aquando da sua realização.

Verifica-se que já existiram projetos onde foi necessário a especificação de novas *Epics* após o início do projeto, ou seja, as equipas conseguem responder às novas necessidades do cliente. Caso não existam *Epics* definidas após o início do projeto verifica-se que o processo de elicitação inicial dos requisitos foi suficiente para atender a todas as necessidades do cliente.

### 7.3.2 Objetivo 2. Melhorar o processo de transformação das necessidades do cliente em requisitos

Q3. As *Epics* são definidas no início do projeto?

Tabela 8 - M3. Percentagem de *Epics* definidas no início do projeto

| ***Projeto*** | ***M3. Percentagem de Epics definidas no início do projeto*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 83.3% |
| ***Projeto 2 – Plataforma Web*** | 100% |
| ***Projeto 3 – App guia turístico*** | 100% |
| ***Projeto 4 – Loja online*** | 91.7% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 91.7% |
| | |
| ***Média*** | 93.3% |

Q4. As *Epics* são divididas em *User Stories* menores?

Tabela 9 - M4. Percentagem de *Epics* que foram refinadas em *User Stories* menores

| ***Projeto*** | ***M4. Percentagem de Epics que foram refinadas em User Stories menores*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 83.3% |
| ***Projeto 2 – Plataforma Web*** | 0% |
| ***Projeto 3 – App guia turístico*** | 16.7% |
| ***Projeto 4 – Loja online*** | 0% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 40% |
| ***Objetivo*** | 100% |

O processo de transformação das necessidades do cliente em requisitos é fulcral para o sucesso da implementação. Caso as necessidades sejam mal percebidas ou mal especificadas a implementação não irá de acordo ao que o cliente necessita. Desta forma deve existir a garantir que este processo funciona o mais corretamente possível.

A definição das *Epics* é o primeiro passo na transformação das necessidades do cliente para o formato de requisito, para tal devem ser definidas com o máximo de clareza possível de forma a dar um bom suporte à equipa. Como por norma as *Epics* são objetos de trabalho grandes e difíceis de trabalhar em ciclos de *feedback* mais curtos, a equipa refina as *Epics* em *User Stories* menores. Ao refinar uma *Epic*, todas as *User Stories* devem ficar com a referência para a respetiva *Epics*. Assim é possível verificar qual o estado de implementação de uma *Epic* tendo como base o estado das *User Stories* relacionadas a si.

Na Tabela 8 em apenas dois projetos são definidas a totalidade das *Epics* no início do projeto. Nos restantes essa definição é realizada também no decorrer do projeto o que permite acompanhar as necessidades do cliente.

Foi definido que todas as *Epcis* devem ser refinadas em *User Stories* menores e que deve ser mantida uma relação entre *Epics* e *User Stories*. Contudo, ao analisar a Tabela 9 apenas no "Projeto 5 – App catálogo digital" é verificada essa prática. Desta forma as alterações ao processo permitiram que o objetivo seja cumprido.

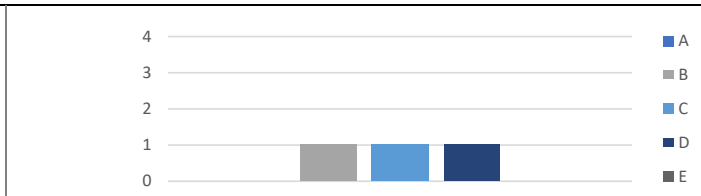
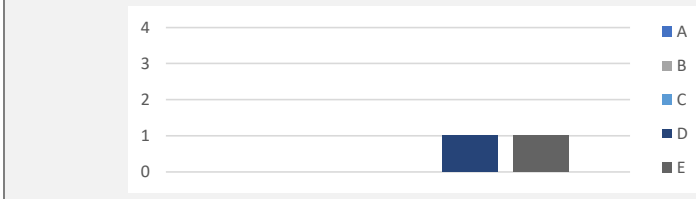
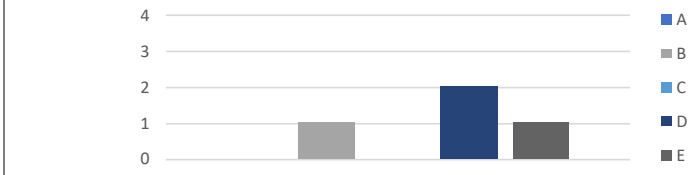
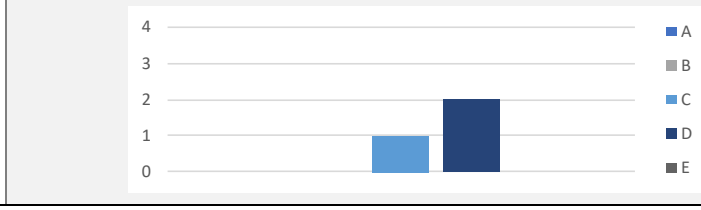
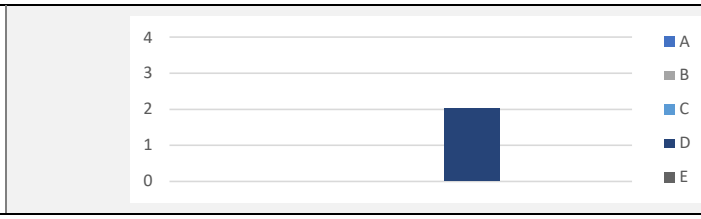
### 7.3.3 Objetivo 3. Melhorar a forma como os requisitos são especificados e documentados na ferramenta de gestão de projeto

Q5. Com que regularidade é feita a cerimónia *Backlog Grooming*?

Os gráficos apresentados na seguinte tabela regem-se pelas seguintes opções:

- **A** – Nunca é realizado (em 0% dos *Sprints*).
- **B** – Raramente no final de cada *Sprint* (em menos de 50% dos *Sprints*).
- **C** – Regularmente no final de cada *Sprint* (em mais de 50% dos *Sprints*, mas não na totalidade).
- **D** – Sempre no final de cada *Sprint* (em 100% dos *Sprints*).
- **E** – A qualquer altura sempre que existe necessidade da sua realização.

Tabela 10 - M5. Regularidade com que é realizada a cerimónia de *Backlog Grooming*

| ***Projeto*** | ***M5. Regularidade com que é realizada a cerimónia de Backlog Grooming*** |
|---|---|
| | *Antes da alteração do processo* |
| ***Projeto 1 – App híbrida com VoIP*** |  |
| ***Projeto 2 – Plataforma Web*** |  |
| ***Projeto 3 – App guia turístico*** |  |
| ***Projeto 4 – Loja online*** |  |
| | *Após a alteração do processo* |
| ***Projeto 5 – App catálogo digital*** |  |

| | |
|---|---|
| ***Objetivo*** | A qualquer altura sempre que existe necessidade da sua realização |

Q6. As *User Stories* apresentam a descrição do cenário de negócio?

Tabela 11 - M6. Percentagem de *User Stories* que apresentam a descrição do cenário de negócio

| ***Projeto*** | ***M6. Percentagem de User Stories que apresentam a descrição do cenário de negócio*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 32% |
| ***Projeto 2 – Plataforma Web*** | 32% |
| ***Projeto 3 – App guia turístico*** | 46% |
| ***Projeto 4 – Loja online*** | 27% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 47.4% |
| ***Objetivo*** | 100% |

Q7. As *User Stories* apresentam a identificação do tipo de plataforma para o qual a *User Story* é direcionada (exemplo: *Back-end*, *Front-end*, *Mobile*, etc.)?

Tabela 12 - M7. Percentagem de *User Stories* que apresentam a identificação do tipo de plataforma

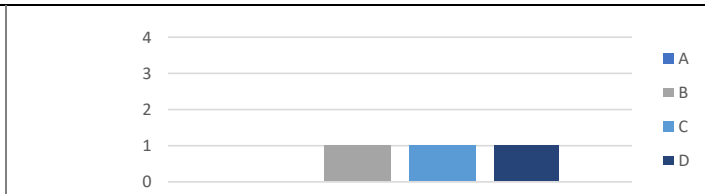
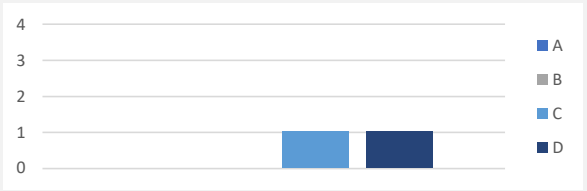
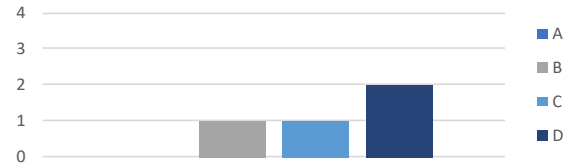
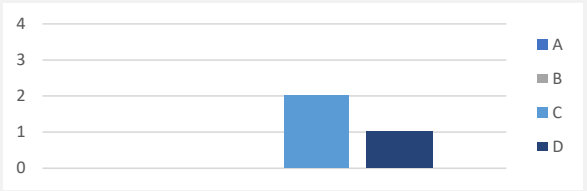
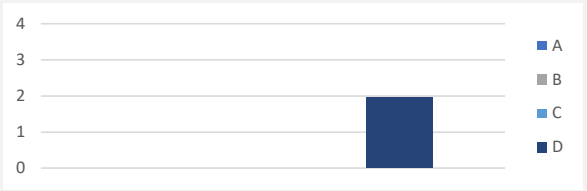
| ***Projeto*** | ***M7. Percentagem de User Stories que apresentam a identificação do tipo de plataforma*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 77% |
| ***Projeto 2 – Plataforma Web*** | 22% |
| ***Projeto 3 – App guia turístico*** | 100% |
| ***Projeto 4 – Loja online*** | 98% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 79.4% |
| ***Objetivo*** | 100% |

Q8. A definição, alteração, exclusão e priorização de *Epics* e *User Stories* do *Product Backlog* ocorre durante o *Backlog Grooming*?

Os gráficos apresentados na seguinte tabela regem-se pelas seguintes opções:

- **A** – Nunca são realizadas durante o *Backlog Grooming*.
- **B** – São realizadas durante o *Backlog Grooming* apenas uma vez no início do projeto.
- **C** – São realizadas durante o *Backlog Grooming* apenas após o início do projeto.
- **D** – São realizadas sempre durante o *Backlog Grooming*.

Tabela 13 - M8. Regularidade com que a definição, alteração, exclusão e priorização de *Epics* e *User Stories* é realizada durante o *Backlog Grooming*

| ***Projeto*** | ***M8. Regularidade com que a definição, alteração, exclusão e priorização de Epics e User Stories é realizada durante o Backlog Grooming*** |
|---|---|
| | *Antes da alteração do processo* |
| ***Projeto 1 – App híbrida com VoIP*** |  |
| ***Projeto 2 – Plataforma Web*** |  |
| ***Projeto 3 – App guia turístico*** |  |
| ***Projeto 4 – Loja online*** |  |
| | *Após a alteração do processo* |
| ***Projeto 5 – App catálogo digital*** |  |
| ***Objetivo*** | São realizadas sempre durante o *Backlog Grooming* |

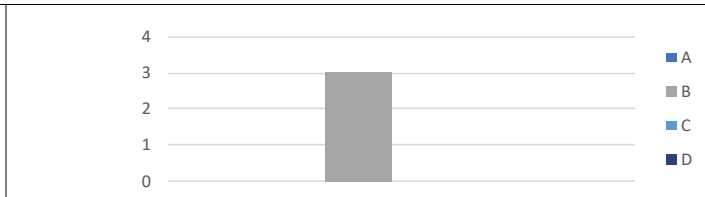
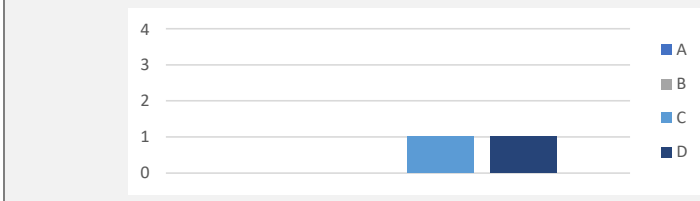
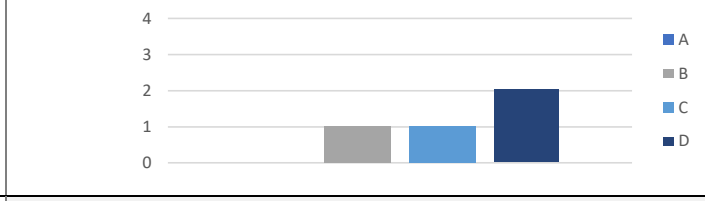
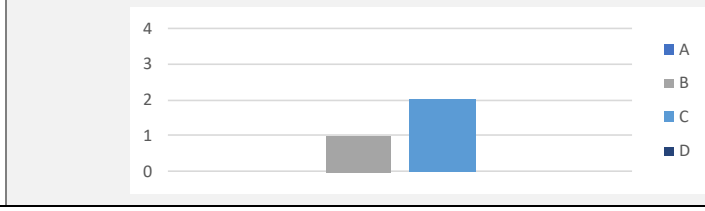
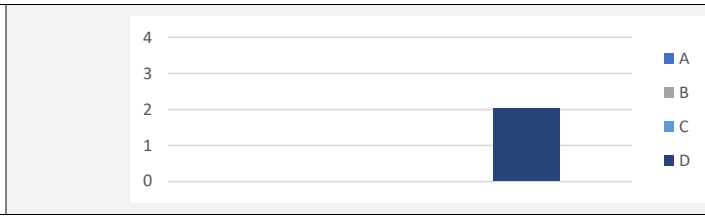
Q9. Todas as *User Stories* e *Epics* a ser implementadas são documentados na ferramenta de gestão e validadas durante *Backlog Grooming*?

Os gráficos apresentados na seguinte tabela regem-se pelas seguintes opções:

- **A** – Nunca são realizadas durante o *Backlog Grooming*.
- **B** – São realizadas durante o *Backlog Grooming* apenas uma vez no início do projeto.

- **C** – São realizadas durante o *Backlog Grooming* apenas após o início do projeto.
- **D** – São realizadas sempre durante o *Backlog Grooming*.

Tabela 14 - M9. Regularidade com que a documentação e validação das *Epics* e *User Stories* acontece durante *Backlog Grooming*

| ***Projeto*** | ***M9. Regularidade com que a documentação e validação das Epics e User Stories acontece durante Backlog Grooming*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** |  |
| ***Projeto 2 – Plataforma Web*** |  |
| ***Projeto 3 – App guia turístico*** |  |
| ***Projeto 4 – Loja online*** |  |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** |  |
| ***Objetivo*** | São realizadas sempre durante o *Backlog Grooming* |

A especificação e documentação dos requisitos ocupa um papel importante na engenharia de requisitos. É importante que a especificação seja clara o suficiente e que sejam documentados corretamente de forma a garantir um fácil acesso e consulta por parte das equipas. O *Backlog Grooming* tem um papel

muito importante na especificação dos requisitos, permite que o que é especificado seja conhecido e validado por toda a equipa. A sua realização em ambientes ágeis é importantíssima e não deve ser colocada de parte, para tal é importante que as equipas sabiam qual o seu funcionamento e os seus objetivos. Não existindo uma descrição o mais completa e clara possível nas *User Stories*, a equipa quando proceder à implementação irá ter dificuldades em perceber o que é realmente para implementar e o porquê da implementação.

Ao analisar o conjunto de métricas apresentadas anteriormente, especialmente as métricas M5, M8 e M9, verifica-se uma grande divergência entre os elementos em relação ao *Backlog Grooming* nos projetos realizados antes da aplicação das alterações ao processo. As divergências podem ter origem na falta de conhecimento por parte nas equipas em relação ao *Backlog Grooming* ou também devido à inexistência de um processo definido pelo qual se devem guiar. No "Projeto 5 – App catálogo digital" o *feedback* obtido é unanime e aproxima-se muito do pretendido, já que no final de cada *Sprint* a equipa realizou o *Backlog Grooming* de forma a deixar o *Product Backlog* o mais atualizado possível. Também no "Projeto 5 – App catálogo digital" todas as operações de gestão e validação das *Epics* e *User Stories* foram realizadas durante o *Backlog Grooming*, o que permitiu cumprir os objetivos pretendidos.

A divergência e inconsistência encontrada na realização do *Backlog Grooming* nos projetos realizados antes da aplicação das alterações ao processo certamente proporcionaram a baixa percentagem de *User Stories* com descrição do cenário de negócio. No "Projeto 5 – App catálogo digital" a realização do *Backlog Grooming* decorreu dentro do normal tal como é previsto no processo. Desta forma foi possível alcançar o objetivo sendo definida uma descrição para cada *User Story* existente. O mesmo se aplica à identificação visual do tipo de plataforma da *User Story*. Apesar de em projetos anteriores às alterações ao processo já serem apresentados valores próximos do objetivo definido, no "Projeto 5 – App catálogo digital" esse objetivo foi alcançado. A identificação visual do tipo de plataforma pode não ser algo crítico na especificação dos requisitos, mas permite que ao consultar as *boards* da ferramenta de gestão de projeto exista a possibilidade de identificar visualmente onde se estão a concentrar os esforços do desenvolvimento. De certa forma também ajuda na melhoria da especificação dos requisitos.

### 7.3.4 Objetivo 4. Reduzir o número de inconsistências entre os requisitos definidos e a sua respetiva implementação

Q10. As *Epics* apresentam a descrição do que é pretendido de acordo com as necessidades do cliente?

Tabela 15 - M10. Percentagem de *Epics* que apresentam descrição

| ***Projeto*** | ***M10. Percentagem de Epics que apresentam descrição*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 0% |
| ***Projeto 2 – Plataforma Web*** | 0% |
| ***Projeto 3 – App guia turístico*** | 0% |
| ***Projeto 4 – Loja online*** | 0% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 20% |
| ***Objetivo*** | 100% |

Q11. As *User Stories* são definidas com clareza suficiente?

Tabela 16 - M11. Clareza com que são definidas as *User Stories*

| ***Projeto*** | ***M11. Clareza com que são definidas as User Stories*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 45.2% |
| ***Projeto 2 – Plataforma Web*** | 30.9% |
| ***Projeto 3 – App guia turístico*** | 49.6% |
| ***Projeto 4 – Loja online*** | 44.9% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 54.1% |
| ***Objetivo*** | 95% |

A clareza dos requisitos está diretamente relacionada à qualidade da implementação, para tal é necessário garantir o máximo de clareza possível no memento da especificação, seja de uma *Epic* ou de uma *User Story*.

No caso das *Epics* o atributo mais relevante é a sua descrição e a relação com as *User Stories*. Antes aplicação das alterações ao processo não existia qualquer *Epic* com descrição, desta forma foi definido que todas as *Epics* devem apresentar uma descrição. Esse objetivo foi alcançado no Projeto 5.

A clareza das *User Stories* é calculada através de um conjunto de métricas, o seu cálculo é realizado da seguinte forma:

$$M11 = \frac{(M6 + M7 + M12 + M13 + M14)}{5}$$

Ao analisar a Tabela 16 é possível verificar se as *User Stories* estão a ser especificadas de acordo com o processo existente ou não. Nesta fase da otimização foi definido que as *User Stories* de um projeto devem apresentar pelo menos 95% na clareza com que são definidas. Apenas o "Projeto 5 – App catálogo digital" consegue atingir esse nível de satisfação, os restantes projetos realizados antes da aplicação das alterações ao processo estão distantes desse objetivo. Não existindo clareza nos requisitos irão certamente aparecer incoerências entre o que o cliente pretende e o que é realmente implementado através de pressupostos criados devido à falta de informação. Foram encontrados vários problemas e melhorias necessárias em projetos com baixos níveis clareza, tendo sido prolongada a data de entrega na sua maioria.

### 7.3.5 Objetivo 5. Garantir que todas as *User Stories* oferecem suporte às *Epics* definidas

Q12. Todas as *User Stories* são refinadas através de *Epics*?

Tabela 17 - M12. Percentagem de *User Stories* que apresentam a relação com uma *Epic*

| ***Projeto*** | ***M12. Percentagem de User Stories que apresentam a relação com uma Epic*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 18.7% |
| ***Projeto 2 – Plataforma Web*** | 0% |
| ***Projeto 3 – App guia turístico*** | 1.5% |
| ***Projeto 4 – Loja online*** | 0% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 24% |
| ***Objetivo*** | 100% |

É importante garantir que as *User Stories* são refinadas através das *Epics*. Seguindo este pressuposto não irão existir *User Stories* perdidas no *Product Backlog* e a equipa irá saber a origem de cada uma das delas. Existindo a relação entre as *Epics* e as *User Stories* será possível não só verificar o progresso das *Epics* como também verificar se essas mesmas *User Stories* estão a oferecer suporte ao que é pretendido. Foi definido que todas as *User Stories* deve apresentar relação com um *Epic*. Antes de aplicadas as alterações ao processo essa prática era muito pouco considerada pelas equipas e os projetos apresentam baixos níveis nessa na relação entre *Epics* e *User Stories*. No "Projeto 5 – App catálogo digital" todas as *User Stories* apresentam relação com um *Epic* e o objetivo pretendido foi alcançado.

### 7.3.6 Objetivo 6. Melhorar a forma como os critérios de conclusão dos requisitos são definidos

Q13. As *User Stories* apresentam a descrição dos critérios de aceitação?

Tabela 18 - M13. Percentagem de *User Stories* que apresentam a descrição dos critérios de aceitação

| ***Projeto*** | ***M13. Percentagem de User Stories que apresentam a descrição dos critérios de aceitação*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 0% |
| ***Projeto 2 – Plataforma Web*** | 0% |
| ***Projeto 3 – App guia turístico*** | 0% |
| ***Projeto 4 – Loja online*** | 0% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 20% |
| ***Objetivo*** | 90% |

Q14. As *Users Stories* apresentam uma lista de tarefas que definem a sua funcionalidade?

Tabela 19 - M14. Percentagem de *User Stories* que apresentam uma lista de tarefas

| ***Projeto*** | ***M14. Percentagem de User Stories que apresentam uma lista de tarefas*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 99% |
| ***Projeto 2 – Plataforma Web*** | 100% |
| ***Projeto 3 – App guia turístico*** | 100% |
| ***Projeto 4 – Loja online*** | 100% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 99.8% |
| ***Objetivo*** | 100% |

Os critérios de conclusão e atributos de qualidade são aspetos muito importantes na especificação de uma *User Story*. A sua inexistência tem um grande impacto na forma como as equipas entendem se uma *User Story* está implementada ou não e também na forma como são realizados os testes ao sistema. Com a existência de margem para a criação de pressupostos por parte das equipas os sistemas desenvolvidos podem ser criados com um comportamento que poderá não respeitar o pretendido pelos clientes. De forma a garantir o total entendimento das *User Stories*, as equipas devem criar, para cada *User Story*, um conjunto de tarefas que representam a funcionalidade a ser desenvolvida na *User Story*.

Foi definido que as *User Stories* devem apresentar uma lista de critérios de aceitação e uma lista de tarefas que representem a funcionalidade desenvolvida. Nesta fase da otimização foi definido um objetivo de 90% para a lista de critérios de aceitação e 100% para a lista de tarefas. Nos projetos realizados antes da aplicação das alterações ao processo não era feita qualquer definição dos critérios de aceitação e atributos de qualidade nas *User Stories*, já relativamente às tarefas o objetivo é cumprido em quase todos

os projetos. Por fim no "Projeto 5 – App catálogo digital" ambos os objetivos são cumpridos, já que todos as *User Stories* apresentam uma listagem de critérios de aceitação e uma lista de tarefas.

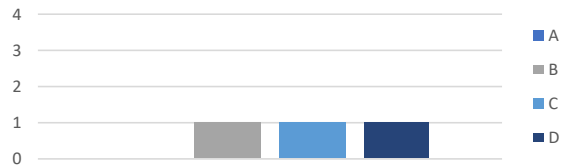
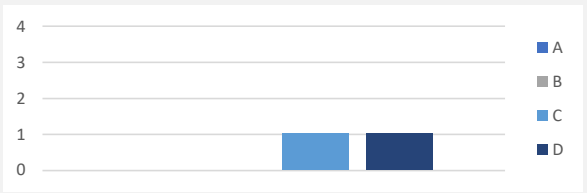
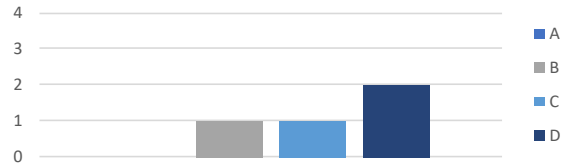
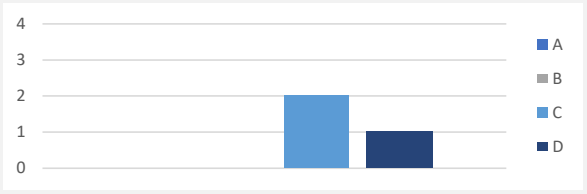
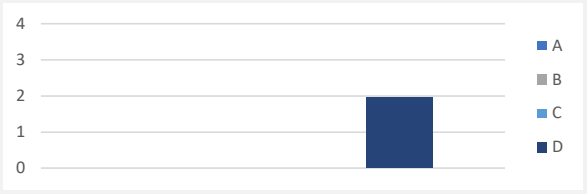
### 7.3.7 Objetivo 7. Melhorar a forma como é feita a gestão de alterações das *Epics* e *User Stories*

Q15. As *Epics* são definidas, alteradas e excluídas pelo *Product Owner* sendo cada alteração discutida com a equipa?

Os gráficos apresentados na seguinte tabela regem-se pelas seguintes opções:

- **A** – Nunca são realizadas durante o *Backlog Grooming*.
- **B** – São realizadas durante o *Backlog Grooming* apenas uma vez no início do projeto.
- **C** – São realizadas durante o *Backlog Grooming* apenas após o início do projeto.
- **D** – São realizadas sempre durante o *Backlog Grooming*.

Tabela 20 - M8. Regularidade com que a definição, alteração, exclusão e priorização de *Epics* e *User Stories* é realizada durante o *Backlog Grooming*

| ***Projeto*** | ***M8. Regularidade com que a definição, alteração, exclusão e priorização de Epics e User Stories é realizada durante o Backlog Grooming*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** |  |
| ***Projeto 2 – Plataforma Web*** |  |
| ***Projeto 3 – App guia turístico*** |  |
| ***Projeto 4 – Loja online*** |  |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** |  |
| ***Objetivo*** | São realizadas sempre durante o *Backlog Grooming* |

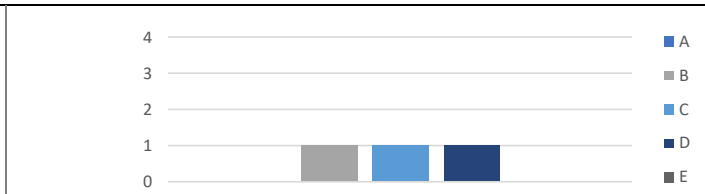
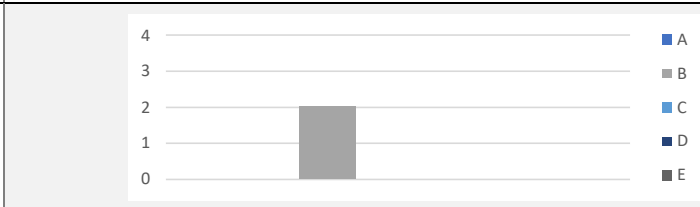
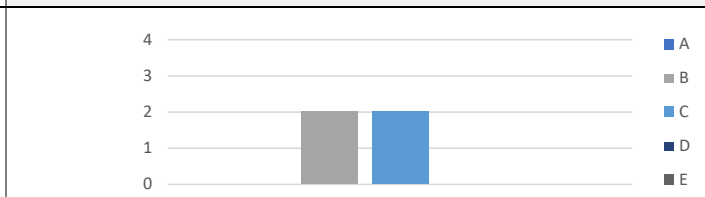
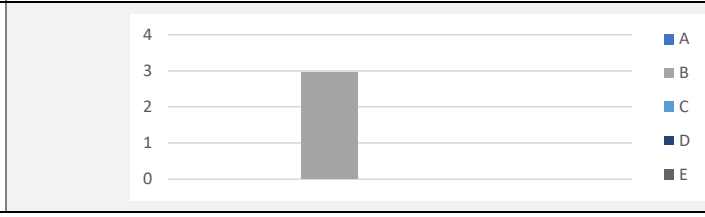
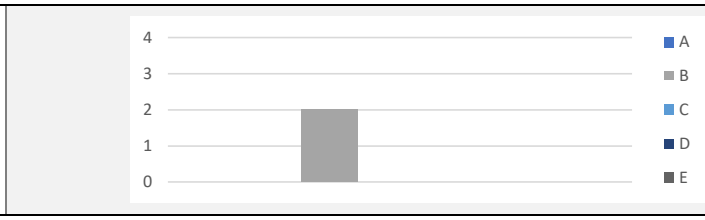
Q16. Com que regularidade as alterações nas *Epics* têm influência em outras *Epics* já definidas?

Os gráficos apresentados na seguinte tabela regem-se pelas seguintes opções:

- **A** – Nunca (0% das vezes que existem alterações).
- **B** – Raramente (até 25% das vezes que existem alterações).

- **C** – Às vezes (até 50% das vezes que existem alterações).
- **D** – Muitas vezes (até 75% das vezes que existem alterações).
- **E** – Sempre (100% das vezes que existem alterações).

Tabela 21 - M15. Regularidade com que as alterações numa *Epic* influenciam a definição de outras *Epics*

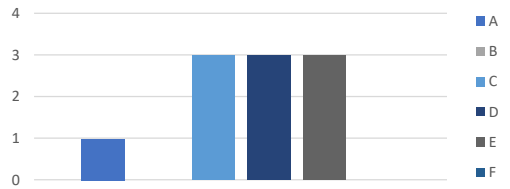
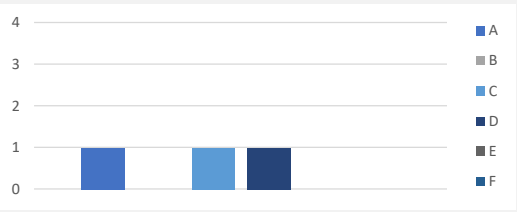
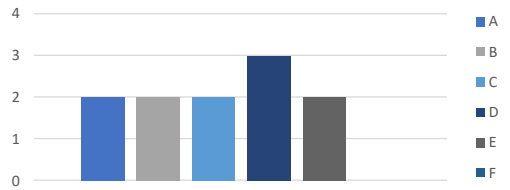
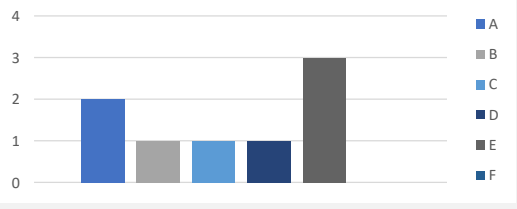
| ***Projeto*** | ***M15. Regularidade com que as alterações numa Epic influenciam a definição de outras Epics*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** |  |
| ***Projeto 2 – Plataforma Web*** |  |
| ***Projeto 3 – App guia turístico*** |  |
| ***Projeto 4 – Loja online*** |  |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** |  |
| ***Objetivo*** | Raramente (até 25% das vezes que existem alterações) |

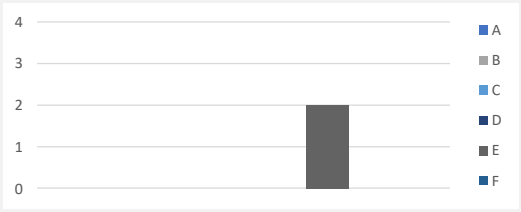
Q17. Quais são os motivos que provocam alterações nas *Epics*?

Os gráficos apresentados na seguinte tabela regem-se pelas seguintes opções:

- **A** – Elicitação realizada inicialmente foi incompleta ou insuficiente.
- **B** – A definição da *Epic* não foi clara o suficiente.
- **C** – Os requisitos foram mal percebidos pela equipa.
- **D** – Surgiram necessidades complementares no âmbito do projeto.
- **E** – Surgiram dificuldades técnicas não previstas pela equipa.
- **F** – Outro.

Tabela 22 - M16. Motivos que provocam alterações nas *Epics*

| ***Projeto*** | ***M16. Motivos que provocam alterações nas Epics*** |
|---|---|
| | *Antes da alteração do processo* |
| ***Projeto 1 – App híbrida com VoIP*** |  |
| ***Projeto 2 – Plataforma Web*** |  |
| ***Projeto 3 – App guia turístico*** |  |
| ***Projeto 4 – Loja online*** |  |
| | *Após a alteração do processo* |

| *Projeto* | *M16. Motivos que provocam alterações nas Epics* |
|---|---|
| ***Projeto 5 – App catálogo digital*** |  |

Como referido anteriormente o *Backlog Grooming* é uma peça fundamental na gestão e desenvolvimento de requisitos. De forma a existir um melhor controlo na gestão de alterações dos requisitos é importante a correta realização desta cerimónia e que seja realizada regularmente. O motivo pelo qual as *Epics* são alteradas pode ser muito variado, segundo o histórico existente é possível verificar que problemas ligados à incompleta ou insuficiente elicitação dos requisitos e à pouca clareza com que são definidos tiveram um impacto bastante negativo nos projetos.

Nos projetos realizados antes da aplicação das alterações ao processo, foi obtido *feedback* com muitas divergências em relação ao *Backlog Grooming*, o que leva a crer que a cerimónia não é realizada corretamente por parte das equipas. Já no "Projeto 5 – App catálogo digital" não se verifica o mesmo, onde é unanime que a gestão das *Epics* e *User Stories* é realizada no *Backlog Grooming*. É importante garantir não só que as alterações são bem geridas, mas também que caso sejam afetados outros requisitos que sejam tomadas as medidas apropriadas em todos os requisitos afetados. Para tal a gestão dos requisitos e respetivas alterações deve ser feita durante o *Backlog Grooming* de forma a garantir que toda a equipa fica a par do que irá ser alterado e valida as respetivas alterações. Garantindo essa prática, o número de problemas associados á elicitação e à clareza serão reduzidos. Com as alterações feitas ao processo este objetivo foi conseguido.

### 7.3.8 Objetivo 8. Manter a rastreabilidade dos requisitos ao longo do desenvolvimento do projeto

Q18. As *User Stories* definidas têm referência para a *Epic* através da qual foram definidas?

Tabela 23 - M12. Percentagem de *User Stories* que apresentam a relação com uma *Epic*

| *Projeto* | *M12. Percentagem de User Stories que apresentam a relação com uma Epic* |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 18.7% |
| ***Projeto 2 – Plataforma Web*** | 0% |
| ***Projeto 3 – App guia turístico*** | 1.5% |
| ***Projeto 4 – Loja online*** | 0% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 24% |
| ***Objetivo*** | 100% |

A rastreabilidade entre as *Epics* e as *User Stories* é muito importante para uma boa gestão do *Product Backlog*. Caso não seja garantida, será muito mais difícil para as equipas conseguirem perceber o que já foi ou não especificado e implementado, para conseguirem realizar um mapeamento hierárquico dos requisitos e também para perceber se uma determinada *Epic* já está concluída ou não. Nos projetos realizados antes da aplicação das alterações ao processo a relação entre *Epics* e *User Stories* é muito baixa. Desta forma foi definido que todas as *User Stories* devem ter relação com a *Epic* de onde foram refinadas. No "Projeto 5 – App catálogo digital" essa prática é verificada em 100% dos casos, permitindo assim atingir o objetivo pretendido.

### 7.3.9 Objetivo 9. Melhorar a forma como é feita a gestão dos tempos de execução das *Epics* e das *User Stories*

Q19. As *User Stories* apresentam a descrição do tempo gasto, em horas, na sua implementação?

Tabela 24 - M17. Percentagem de *User Stories* que apresentam a descrição do tempo gasto na implementação

| ***Projeto*** | ***M17. Percentagem de User Stories que apresentam a descrição do tempo gasto na implementação*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 66% |
| ***Projeto 2 – Plataforma Web*** | 81% |
| ***Projeto 3 – App guia turístico*** | 82% |
| ***Projeto 4 – Loja online*** | 80% |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 100% |
| | |
| ***Média*** | 81.8% |
| ***Objetivo*** | 100% |

Q20. Qual o tempo médio em horas gasto para implementar uma *User Story*?

Tabela 25 - M18. Tempo médio em horas para implementar uma *User Story*

| ***Projeto*** | ***M18. Tempo médio em horas para implementar uma User Story*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 2h56 |
| ***Projeto 2 – Plataforma Web*** | 3h07 |
| ***Projeto 3 – App guia turístico*** | 5h12 |
| ***Projeto 4 – Loja online*** | 3h00 |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 2h46 |
| | |
| ***Média*** | 3h24 |

Q21. Qual o tempo médio em horas gasto para implementar uma *Epic*?

Tabela 26 - M19. Tempo médio em horas para implementar uma *Epic*

| ***Projeto*** | ***M19. Tempo médio em horas para implementar uma Epic*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** | 19h05 |
| ***Projeto 2 – Plataforma Web*** | 0h00 |
| ***Projeto 3 – App guia turístico*** | 2h30 |
| ***Projeto 4 – Loja online*** | 0h00 |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** | 17h58 |
| | |
| ***Média*** | 7h55 |

O registo dos tempos de execução das *User Stories* é algo importante na ótica da gestão de projeto, isto porque não só permite perceber o derrape (caso exista) do projeto como também criar histórico que irá auxiliar em futuras estimativas para projetos semelhantes ou com requisitos em comum. Através da existência de rastreabilidade nos requisitos é possível perceber qual o tempo de execução de cada *Epic*, bastando aplicar o somatório dos tempos das *User Stories* a si relacionadas.

O registo dos tempos é algo que veio a ser feito ao longo do tempo, mesmo em projetos realizados anteriormente. Contudo não foi considerado em todos os casos por parte das equipas, isto porque em nenhum dos projetos realizados da aplicação das alterações ao processo existe a totalidade de *User Stories* com o tempo de execução preenchido. Foi definido que todas as *User Stories* devem apresentar o tempo de execução. Por fim no "Projeto 5 – App catálogo digital" já se verifica que a totalidade das *User Stories* contem o tempo de execução.

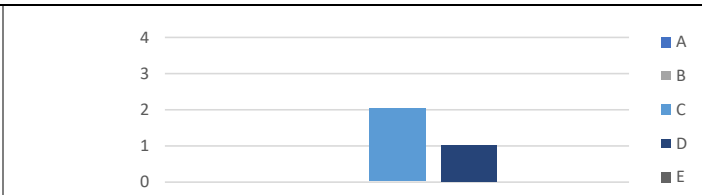
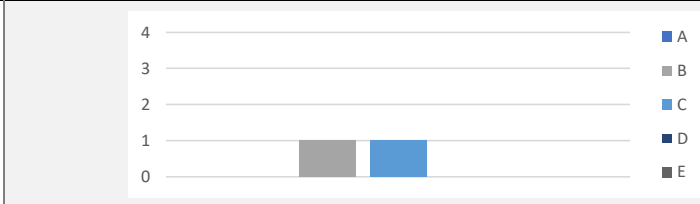
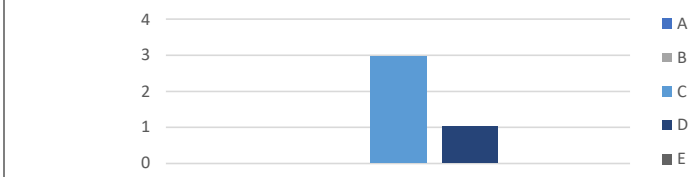
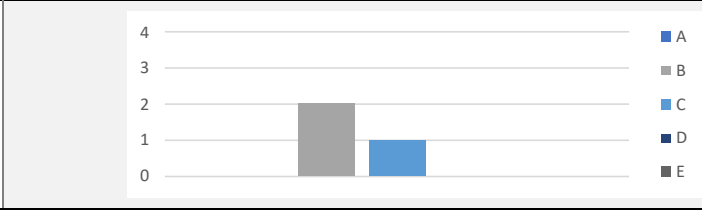
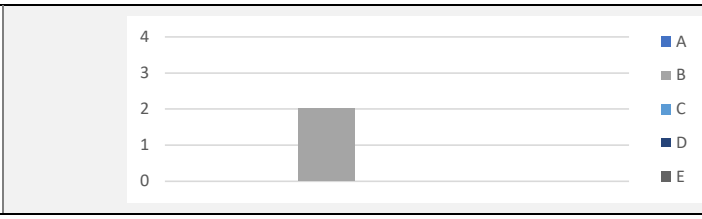
### 7.3.10 Objetivo 10. Medir o quão bem a equipa se concentra nas necessidades do projeto

Q22. Com que regularidade a priorização do *Product Backlog* é afetada devido à não disponibilização de ferramentas ou informações necessárias ao desenvolvimento?

Os gráficos apresentados na seguinte tabela regem-se pelas seguintes opções:

- **A** – Nunca (nunca é afetada por fatores externos).
- **B** – Raramente (até 25% das vezes que existem fatores externos).
- **C** – Às vezes (até 50% das vezes que existem fatores externos).
- **D** – Muitas vezes (até 75% das vezes que existem fatores externos).
- **E** – Sempre (sempre que existem fatores externos).

Tabela 27 - M20. Regularidade com que as alterações numa *Epic* influenciam a definição de outras *Epics*

| ***Projeto*** | ***M20. Regularidade com que as alterações numa Epic influenciam a definição de outras Epics*** |
|---|---|
| *Antes da alteração do processo* | |
| ***Projeto 1 – App híbrida com VoIP*** |  |
| ***Projeto 2 – Plataforma Web*** |  |
| ***Projeto 3 – App guia turístico*** |  |
| ***Projeto 4 – Loja online*** |  |
| *Após a alteração do processo* | |
| ***Projeto 5 – App catálogo digital*** |  |
| ***Objetivo*** | Raramente (até 25% das vezes que existem fatores externos) |

Quando existem fatores externos ao desenvolvimento, tais como ferramentas ou informações necessárias ao desenvolvimento, não deve existir a necessidade de alterar a priorização de forma inesperada devido ao atraso na disponibilização dos mesmos à equipa. Além poder colocar em causa o planeamento realizado também pode comprometer as necessidades mais urgentes do cliente. O objetivo é que este tipo de situação aconteça o menos possível, tendo sido fixado como objetivo em: Raramente (até 25% das vezes que existem fatores externos). Existem algumas divergências nos projetos realizados antes da aplicação das alterações ao processo, mas a maioria aponta elevadas taxas de alteração da priorização quando existem

dependências de fatores externos ao desenvolvimento. Por fim no "Projeto 5 – App catálogo digital", e apesar da priorização ter sido afetada por fatores externos ao desenvolvimento, a percentagem foi baixa e enquadra-se nos objetivos.

## 7.4 Conclusões

Após a avaliação e análise realizadas verifica-se que a grande maioria dos objetivos definidos foram alcançados, tendo sido diminuído em grande parte a quantidade de problemas que os projetos realizados anteriormente à estruturação do processo apresentaram.

Na secção 7.3.1, relativamente ao objetivo de aproximar os requisitos do cliente continua a existir alguma dificuldade em obter *feedback* no final de cada *Sprint*, mas apesar disso foi possível acompanhar as necessidades do cliente tendo sido especificadas *Epics* após o início do projeto. Na secção 7.3.2 o objetivo foi alcançado na totalidade, todas as *Epcis* forma refinadas em *User Stories* menores mantendo a rastreabilidade e oferecendo suporte bidirecional entre requisitos.

Nos projetos realizados anteriormente a este trabalho, verificou-se a existência de divergências no que diz respeito ao *Backlog Grooming* e a como as *Epics* e User Stories foram geridas. Desta forma na secção 7.3.3 foi definido o objetivo de melhorar a forma como os requisitos são especificados e documentados na ferramenta de gestão de projeto. Apesar de, no projeto realizado após a estruturação do processo, a cerimónia se continuar a realizar no final de cada *Sprint* e não sempre que necessário, foi uma informação fornecida de forma unânime por parte da equipa. Isto leva a concluir que os membros saberiam efetivamente qual o contexto no qual foi realizado o desenvolvimento e gestão dos requisitos. Além disso foi alcançado o objetivo de todas as *User Stories* apresentarem uma descrição do cenário de negócio, que é muito importante já que permite fazer a articulação entre o negócio e a parte técnica do requisito.

Na secção 7.3.4 foi pretendido reduzir o número de inconsistências ente os requisitos definidos e a sua respetiva implementação, para tal foi necessário garantir um maior nível de clareza nas User Stories. Este nível foi alcançado após a estruturação do processo no "Projeto 5 – App catálogo digital". Além disso foi alcançado o objetivo da secção 7.3.5 que pretende garantir que todas as *User Stories* oferecem suporte às *Epics* definidas através da relação criada entre ambas na ferramenta de gestão de projeto.

O objetivo da secção 7.3.6 remete melhorar a forma como os critérios de conclusão dos requisitos são definidos. Após a estruturação do processo, foi possível garantir que as *User Stories* apresentam critérios de conclusão, deixando de ficar ao critério dos elementos da ST decidir se a *User Story* está ou não implementada. Isto porque, existindo um conjunto de critérios de aceitação e uma lista de tarefas será muito mais claro saber o que é realmente necessário implementar.

Na secção 7.3.7 foi pretendido melhorar a forma como é feita a gestão de alterações das *Epics* e *User Stories*. Como já referido na presente secção, os projetos realizados anteriormente à estruturação apresentam grandes divergências relativamente a cerimónia *Backlog Grooming*. Também nestes projetos são reportados vários problemas relacionados a elicitação, especificação e clareza dos requisitos. Com o trabalho realizado o número de problemas diminuiu consideravelmente, sendo reportadas apenas problemas reportados com dificuldades técnicas não previstas pela ST. O facto de na secção 7.3.8 o objetivo ser manter

a rastreabilidade dos requisitos ao longo do desenvolvimento do projeto pode ter influenciado na eficiência na gestão de alterações. Isto porque tal como foi possível fornecer suporte ás *Epics* através das *User Stories*, também foi possível criar rastreabilidade bidirecional entre os requisitos, sendo possível perceber de imediato em toda a cadeia de informação que se esta a mexer. Além disso é possível perceber facilmente o que já foi ou não especificado e caso tenha sido qual o seu estado, auxiliando as equipas a perceber se uma *Epic* já foi ou não totalmente refinada ou implementada, isto com base nas *User Stories* a si relacionadas.

A criação de histórico é um objetivo da organização, isto de forma a permitir, no futuro, realizar estimativas mais acertadas de acordo com projetos e requisitos semelhantes. Na secção 7.3.9 foi definido o objetivo de melhorar a forma como é feita a gestão dos tempos de execução das *Epics* e das *User Stories*. Para tal foi necessário garantir que a ST registe os tempos necessários para o desenvolvimento das *User Stories*. Esta prática não era cumprida na integra pelas equipas, existindo *User Stories* que não contêm essa informação. Este objetivo foi alcançado após a estruturação do processo no "Projeto 5 – App catálogo digital", onde todas as *User Stories* apresentam o acumulado de horas necessário para o seu desenvolvimento.

Por fim na secção 7.3.10 foi definido que seria necessário medir o quão bem a equipa se concentra nas necessidades do projeto. Para tal foi necessário perceber se existem alterações não previstas na priorização devido à falta de informação ou atraso na disponibilização de ferramentas necessárias ao projeto. Apesar da alteração da priorização ser algo que acontece regularmente nos projetos da organização, é isso não ser um problema tendo como base os princípios das principais abordagens ágeis de desenvolvimento de *software*. A alteração da priorização devido ao atraso na disponibilização de ferramentas ou informações necessárias ao desenvolvimento pode atrasar a calendarização e colocar em risco os objetivos do negócio em questões temporais. Mesmo no projeto realizado após a estruturação do processo o problema continua a acontecer, apesar de ser em menor percentagem.

Pode ser concluído que as alterações ao processo representam uma melhoria para a organização e que será possível melhorar o nível de maturidade da organização tanto na gestão e desenvolvimento de requisitos como também no PDS devido à documentação auxiliar desenvolvida neste trabalho. A apreciação geral aos resultados foi positiva, no entanto a avaliação poderia ter incluído aplicação da estruturação realizada neste em mais projetos. Apenas com um único projeto para comparação não foi possível sujeitar o processo a vários ambientes de desenvolvimento e consequentemente a um maior número de riscos.

# 8 Atualização do Processo e Formação

As alterações ao processo refletiram-se em melhorias, desta forma é necessário proceder à atualização do processo existente com as novas estruturas propostas e realizar a formação necessária para que os colaboradores consigam trabalhar com o processo.

Na secção 8.1 será abordado quais as atualizações realizadas com base neste trabalho e também as atualizações que motivou na organização. Em seguida na secção 8.2 é referida a importância da formação dos colaboradores e quais as formações realizadas no âmbito deste trabalho. Por fim na secção 8.3 são apresentadas as considerações finais da atualização do processo e formação e feita a correspondência com alguns dos objetivos e práticas genéricas do CMMI.

## 8.1 Atualização do Processo

Sempre que são realizadas alterações ao processo e que é mostrado que representa uma melhoria ou mais-valia para a organização, deve-se proceder à atualização dos processos efetuados pela alteração. O processo de atualização deve incluir a atualização da documentação dos processos com novas versões e também a formação dos colaboradores (em mais detalhe na secção 8.2).

Como referido na secção 5.1 a Crossing Answers não tem o seu PDS devidamente definido e documentado de forma a que qualquer colaborador o consiga consultar e esclarecer as suas dúvidas. O primeiro passo passou por criar a primeira versão do PDS da Crossing Answers e os seus respetivos anexos. Foram desenvolvidos vários anexos que pretendem fornecer esclarecimentos adicionais a quem está a consultar os processos, por exemplo em matérias como o Scrum e o Kanban. Além disso foram desenvolvidos *templates* para a especificação de *Epics* (Anexo C) e de *User Stories* (Anexo D). Nos *templates*, além da contextualização, são também apresentados exemplos práticos para cada um dos respetivos atributos. O processo de requisitos definido neste trabalho irá ser utilizado internamente na organização e será comtemplado na estrutura do PDS definido para utilização na Crossing Answers.

Além da criação de documentos e *templates* que irão dar suporte ás equipas de desenvolvimento daqui para a frente, ouve uma outra atualização que teve origem neste trabalho, foi ela a alteração da ferramenta de gestão de projetos usada internamente. A ferramenta usada anteriormente e aquando a realização deste trabalho requeria bastantes configurações no início de cada projeto uma vez que não permitia a criação de *templates* para utilização comum nos projetos. Além disso não disponibiliza qualquer ferramenta que permite criar relatórios de forma a medir determinados aspetos do projeto, apenas fornece um conjunto de ficheiros com toda a informação do projeto e que posteriormente podem ser tratados para obter métricas (como foi o caso deste trabalho).

Foi necessário escolher uma ferramenta de gestão de projetos que permitisse a configuração inicial das *boards* para as equipas usarem sem qualquer tipo de trabalho repetitivo de projeto para projeto e que

também permitisse a definição de métricas e respetiva obtenção de valores se necessidade de desenvolvimentos adicionais (como foi o caso deste trabalho). Foi então escolhida uma nova ferramenta de gestão de projetos com base nesses critérios. A sua escolha não foi da responsabilidade deste trabalho, apenas a motivação para a adoção de algo mais avançado.

## 8.2 Formação

O primeiro objetivo genérico do CMMI sugere que sejam alcançadas e executadas as práticas específicas das áreas de processo abordadas na organização, para tal é muito importante que todos estejam capazes de executar todas as técnicas e cerimónias que o PDS inclui na sua definição. Ainda no CMMI, segundo a prática genérica *GP 2.5 – Treinar pessoas*, é necessário orientar a organização e fornecer níveis adequados de formação para todos os intervenientes nos desenvolvimentos para que seja possível que desempenhem as suas funções com eficiência.

Apesar de não ter sido realizada nenhum *workshop* totalmente dedicado ao processo de requisitos definido no capítulo 6, foi, em ambas as apresentações, dado o destaque necessário ao conjunto de atividades do processo de requisitos, tal como aos artefactos aí apresentados. Isto foi possível porque o processo de requisitos foi definido tendo em conta o conjunto de cerimónias e artefactos disponibilizados pelo Scrum.

De forma a respeitar os princípios referidos foram realizados *workshops* e criados materiais de apoio para os colaboradores, como referido na secção 8.1. Ao longo da secção 7.3 foram encontradas muitas discordâncias entre os colaboradores no que diz respeito as métricas relativas as cerimónias. Desta forma foi realizado um *workshop* orientado ao Scrum e um outro orientado ao Kanban.

### 8.2.1 *Workshop* de Scrum

O primeiro workshop realizado foi o de Scrum. Na parte teórica do *workshop* foi realizada uma apresentação onde foram abordados vários tópicos relacionados com o Scrum e também com o ágil de forma geral. A agenda da apresentação foi a seguinte:

- Ágil e os seus valores e princípios.
- Problemas e desafios do ágil.
- Metodologias ágeis (*Kanban, Extreme Programming e Scrum*).
- Scrum e as suas características.
- Valores do Scrum (Compromisso, foco, respeito e coragem).
- Processo e ciclo do Scrum (*Pre-game*, Desenvolvimento e *Post-game*).
- Papeis do Scrum (PO, SM, ST).
- Principais cerimónias do Scrum (*Backlog Grooming*, *Sprint Planning*, *Daily Scrum Meeting*, *Sprint Review*, *Sprint Retrospective*).
- Principais artefactos do Scrum (*Epics*, *User Stories*, *Product Backlog*, *Sprint Backlog*, *Sprint Burndown Chart*).

Após a parte teórica foi realizado uma atividade prática com os participantes no *workshop*. A atividade realizada consistiu na realização do jogo "The Name Game" (Kniberg, 2011). Este jogo é dividido em duas partes, uma primeira parte relativa ao jogo propriamente dito (cerca de 20 minutos) e a segunda parte uma discussão final de forma a obter conclusões em relação aos acontecimentos do jogo (10 a 20 minutos).

Para a realização do jogo foi necessário reunir alguns materiais, tais como canetas, *post-its*, quadro/painel e um cronometro. Ainda antes de formar as equipas (6-7 elementos), foram feitas as seguintes questões aos participantes:

1. Quanto tempo se demora a escrever um nome?
2. Quanto tempo se demorar a escrever sete nomes?
3. Quais os fatores que podem influenciar os tempos estimados?

O conjunto de respostas fornecidas ás anteriores perguntas foi devidamente anotado no quadro existente e foram apresentadas as regras gerais do jogo, são elas:

- Cada grupo vai ter um programador sendo os restantes elementos clientes.
- Cada cliente tem um post-it em branco e pretende ver o seu nome escrito nele, e para tal necessita do programador para o escrever.
- Quando o cliente receber o post-it de volta com o nome completo deve anotar o tempo gasto.
    - Caso o nome esteja incorreto deve devolver o post-it ao programador.

Para a primeira ronda foram definidas um conjunto de políticas e regras adicionais ás definidas anteriormente. As políticas da empresa são regidas pelos princípios de não deixar o cliente à espera e que quanto mais cedo se começar mais cedo se vai terminar. De forma cumprir as políticas da empresa é necessário seguir as seguintes regas:

- Quando o cronometro iniciar os clientes, em simultâneo, dão o cartão ao programador.
- O programador vai escrever a primeira letra de cada nome no respetivo cartão, depois a segunda letra de cada nome e assim sucessivamente até terminar.
- Quando terminar entrega ao cliente para ele escrever o tempo que demorou.
- O programador pode perguntar o que quiser e o cliente pode apenas responder verbalmente.

Após o término da primeira ronda foram apontados os tempos no quadro e verificou-se uma enorme frustração por parte dos participantes uma vez que os tempos que estimaram inicialmente não foram nem de perto os que realmente se verificaram após esta primeira ronda.

Não questionado nem refletindo sobre os resultados foi apresentada a segunda ronda, com um novo conjunto de políticas e regras. Nesta ronda as políticas da empresa são regidas pelos princípios de limitar a

uma unidade de trabalho em simultâneo e a atender um cliente de cada vez. De forma cumprir as políticas da empresa é necessário seguir as seguintes regas:

- Quando o cronometro iniciar apenas um cliente dá o cartão ao programador.
- Só se pode começar a escrever o segundo nome quando o primeiro for terminado e assim sucessivamente.
- O programador decide quando avança para o cliente seguinte.
- O cliente deve apontar o tempo inicial e o tempo final.

No final da segunda ronda, e após o registo dos tempos no quadro, os participantes mostraram-se muito mais satisfeitos com os resultados obtidos, tanto os que representaram programadores como os que representaram clientes. Os tempos de execução foram semelhantes aos tempos estimados inicialmente, antes de iniciar o jogo.

Em seguida teve início a discussão, numa primeira fase foi questionado quais tinham sido os problemas que levaram aos elevados tempos de execução na primeira ronda do jogo. Todos os intervenientes enumeram como principal problema ter de escrever vários nomes em simultâneo, ou seja, a execução de tarefas em *multitasking*. Após a apresentação de vários problemas ligados ao *multitasking* e dos motivos que levam à sua prática, foram apresentadas um novo conjunto de questões aos participantes de forma a fomentar a discussão, forma elas:

1. Sentimento enquanto cliente e programador em cada uma das rondas?
2. Como foi ser o último cliente na ronda 1 e na ronda 2?
3. O que pensam da afirmação “Se começarmos mais cedo, iremos terminar mais cedo”?
4. Como é influenciado o planeamento do trabalho (Ex. o que sabemos após 10 segundos na ronda 1 e na ronda 2)?
5. Como a qualidade é afetada?

A discussão foi muito produtiva e os participantes gostaram da atividade prática, uma que lhes permitiu perceber num cenário tão simples como escrever um conjunto de nomes as repercussões que o *multitasking* pode provocar quando aplicada sobre as equipas.

### 8.2.2 *Workshop* de Kanban

O segundo workshop realizado foi o de Kanban. Na parte teórica do workshop foi realizada uma apresentação onde foram abordados vários tópicos relacionados com o Kanban, com base na seguinte agenda:

- História do Kanban
- Kanban no desenvolvimento de *software*
- Princípios

- Benefícios
- Desvantagens
- Análise e comparação com o Scrum

Após a parte teórica foi realizado uma atividade prática com os participantes no workshop. A atividade realizada consistiu na realização do jogo "Pizza Game" (Rules & Mast, 2013). Tal como a atividade anterior, este jogo é dividido em duas partes, uma primeira parte relativa ao jogo propriamente dito (cerca de 20 minutos) e a segunda parte uma discussão final de forma a obter conclusões em relação aos acontecimentos do jogo (10 a 20 minutos).

Para a realização do jogo foi necessário reunir alguns materiais, tais como canetas coloridas, post-its de várias cores, folhas A4, tesouras, cola e um cronometro. Para a realização do jogo foi necessário fornecer e clarificar às equipas o seguinte conjunto de regras e restrições:

- Todos os cortes devem ser feitos com uma tesoura (uma tesoura por equipa).
- Um marcador para cada equipa.
- Um "forno" contém no máximo 2 fatias de cada vez (um forno por equipa).
- Cada fatia deve ser cozinhada durante 30 segundos e fica queimada após 45 segundos.
- Qualidade de confeção: muito molho e coberturas bem presas à fatia.

Explicadas as regras foi apresentado o método de "confeção" das pizzas, existindo apenas um tipo de pizza (*Pizza Hawaii*) com os seguintes ingredientes:

- Base com borda externa dobrada para cima (folha colorida).
- Molho de tomate (utilizar caneta).
- 3 pedaços de fiambre (post-its rosa/laranja).
- 3 pedaços de abacaxi (post-its amarelos).

Sem mais qualquer tipo de explicação foi dado início à primeira ronda que teve uma duração de cerca de 5 minutos. Esse tempo não foi anunciado as equipas, não era do seu conhecimento quando é que o jogo iria terminar. Após o término da primeira ronda foi explicado como seriam calculados os resultados através de um sistema de pontuação com as seguintes regras:

- Fatias completas (100% pronta): 10 pontos.
- Penalizações:
    - Base com ou sem molho: -4 pontos por cada.
    - Cobertura: -1 ponto por cada.

As equipas apresentaram pontuação muito negativas com grandes quantidades de desperdício. Foi sugerido que para a segunda ronda tivessem em conta a idealização de um fluxo de trabalho explicito e

visível para todos e que cada estado desse fluxo de trabalho tivesse um limite de *work-in-progress*. Para tal foi fornecido material adicional para as equipas prepararem o seu fluxo de trabalho.

Após terminarem de preparar o fluxo de trabalho teve início a segunda ronda. A quantidade de desperdícios foi muito menor e a produtividade aumentou significativamente, deixando a equipa com muito mais motivação para a execução do trabalho proposto. Por fim foi realizada uma última ronda em que foi introduzido um novo tipo de pizza (*Pizza Speciale*) que consiste no seguinte:

- Base com borda externa dobrada para cima (folha colorida).
- Molho de tomate (folha colorida).
- 7 pedaços de rúcula (post-its verdes).
- Os pedaços de rúcula só podem ser adicionados após a Pizza sair do forno.

Este novo tipo provoca alterações no fluxo de trabalho idealizado anteriormente. Apesar disso as equipas tiveram dificuldade em perceber que seria conveniente atualizar o seu fluxo de trabalho, como tal não o fizeram. Desta forma apenas conseguiam produzir o mesmo tipo de pizza em simultâneo.

Por fim foi dado início à discussão onde foram realizadas questões aos participantes relativas ao seu sentimento durante a atividade e como poderia ser feita a analogia entre esta atividade e o trabalho realizado diariamente na empresa.

## 8.3 Considerações Finais

A criação de documentação relativa aos processos fornece um grande apoio ás equipas no desenvolvimento dos projetos, uma vez que podem ser consultados a qualquer altura. Além disso, toda a informação dos processos está centralizada num único local, eliminando a comunicação verbal dos processos.

Para uma correta utilização dos processos e da documentação criada anteriormente, foi muito importante a realização do processo de formação. Foi possível consolidar e colocar à prova os conhecimentos dos colaboradores em trabalho de equipa, em ciclos de trabalho curtos e com necessidade de melhorar com os erros cometidos. A componente de melhoria contínua precisa de ser melhorada e incentivada nos colaboradores, uma vez que foram encontradas algumas dificuldades nas atividades da secção 8.2.

Além das práticas específicas das áreas de processo de *Desenvolvimento de Requisitos* e *Gestão de Requisitos* do CMMI, foram também satisfeitas algumas práticas genéricas ao longo deste trabalho. Como referido na secção 8.2 a prática genérica *GP 2.5 – Treinar pessoas* foi satisfeita uma vez foi fornecida formação e orientação as equipas para a utilização do Scrum e para que desempenhem as suas funções com eficiência. Adotando o Scrum juntamente com as suas cerimónias e técnicas, foi possível definir uma política organizacional e fornecer valores ágeis como a transparência, a colaboração, o falhar rápido e falhar cedo e também o iterativo e incremental. Todos estes fatores permitiram satisfazer a prática genérica *GP 2.1 – Estabelecer uma política organizacional*. Além dos valores referidos é necessário auxiliar as equipas ágeis, fornecendo clareza na definição, uso e execução das várias técnicas e cerimónias, isto é, a existência

de processos definidos por mais simples que sejam. A definição da primeira versão do PDS permitiu em satisfazer a prática genérica *GP 2.2 Planear o processo*.

Concluindo, a definição e otimização de processos por si só não é suficiente para obter melhorias na organização. É necessário, além das ferramentas, formar os colaboradores para as utilizarem corretamente e além disso motivar a sua utilização mostrando a sua importância através de *workshops* e sessões de *team building*.

# 9 Conclusões

Neste capítulo são apresentadas as principais conclusões obtidas neste trabalho. Para tal na secção 9.1 é feita a síntese do trabalho realizado. Posteriormente, na secção 9.2 são apresentados os principais objetivos deste trabalho e qual a metodologia utilizada para as atingir. Na secção 9.3 são apresentadas as principais contribuições deste trabalho e como foram atingidas. Por fim, na secção 9.4 é apresentada a proposta de trabalhos futuros.

## 9.1 Síntese do Trabalho Realizado

Neste trabalho é pretendido otimizar os processos da organização. Desta forma foi necessário identificar os principais problemas existentes nos processos, isto através da informação existente em retrospetivas e também através da recolha de *feedback* junto dos colaboradores. Ao identificar os principais problemas foi necessário selecionar o que deve ser otimizado, e assim proceder à análise, implementação e validação das alterações realizadas. Por fim os processos são atualizados e os colaboradores formados com base nos novos processos.

Começou-se por analisar os processos utilizados na organização (capítulo 4). Como os mesmos não se encontravam documentados, foi necessário proceder, numa primeira fase, a um levantamento do fluxo de trabalho usado internamente. Para obter mais informação, foram analisadas as retrospetivas existentes e recolhido *feedback* junto dos colaboradores. Desta forma, foram encontrados vários problemas em diferentes áreas ligadas ao desenvolvimento e gestão de projetos, tais como o levantamento e gestão de requisitos, a comunicação com o cliente, a inexistência de processos relativos aos testes, aceitação e validação do *software* e a preparação e escolha de dependências externas. Considerando os problemas identificados, foi decidido que o foco deste trabalho seriam as deficiências processuais com impacto no levantamento e gestão de requisitos.

Para perceber quais os principais pontos a melhorar no processo existente, foi consultado o modelo CMMI, em particular as áreas de processo de *Desenvolvimento de Requisitos* e de *Gestão Requisitos* (capítulo 5). O processo de requisitos definido tem como base as práticas descritas no CMMI, mas também respeita as principais práticas ágeis, mais especificamente as do Scrum que é a abordagem adotada pela organização para o desenvolvimento de projetos de *software* (capítulo 6).

A definição do modelo de avaliação do processo desenvolvido foi realizada usando a abordagem GQM. Para tal foram definidos um conjunto de objetivos, questões e métricas de forma a ser possível analisar o impacto do processo desenvolvido nos projetos que o adotam. O processo foi em seguida aplicado a um projeto, tendo sido recolhidos e analisados os indicadores de desempenho identificados na análise GQM, o que revelou melhorias em alguns dos aspetos relevantes (capítulo 7).

Por fim, foram realizadas ações de formação e divulgação de forma a disseminar o novo processo por toda a organização (capítulo 8).

## 9.2 Análise do Cumprimento dos Objetivos

Os objetivos deste trabalho têm principal foco na otimização dos processos da organização. Como foi possível verificar não existiam processos definidos na organização, desta forma também foi necessário proceder a algumas atividades de definição e documentação, nomeadamente do PDS. Desta forma foram definidos os seguintes objetivos para este trabalho (secção 1.2):

1. Realização de um levantamento exaustivo dos processos existentes na organização no que diz respeito ao desenvolvimento de *software*.
2. Identificação dos principais problemas encontrados nos processos da organização.
3. Estruturação dos processos que apresentaram mais problemas na organização.
4. Criação de documentação e *templates* para auxiliar o uso dos processos definidos.
5. Aplicação e avaliação do processo após a estruturação em projetos da organização e, no caso de terem existido melhorias, atualização do mesmo e formação aos colaboradores.

Os objetivos 1 e 2 foram satisfeitos com o trabalho descrito no capítulo 4, onde foi realizada uma avaliação da situação inicial. Nesta avaliação, que se procedeu a seguir a um levantamento do processo e fluxos de trabalho existentes na organização, foi possível identificar os principais problemas nos processos existentes. Do conjunto de problemas identificados foi necessário escolher uma área para proceder à otimização. A opção recaiu sobre o desenvolvimento e gestão de requisitos.

Relativamente ao objetivo 3, foi realizado um estudo do modelo CMMI, em particular no que diz respeito às áreas de processo de gestão e desenvolvimento de requisitos. Estas áreas de processo foram mapeadas em métodos ágeis de desenvolvimento de *software* de forma preparar a estruturação do processo (tal como pode ser consultado no capítulo 5). Desta forma, foi criada uma base para proceder à estruturação do processo de requisitos, onde foram consideradas as práticas específicas das áreas de processo *Desenvolvimento de Requisitos* e *Gestão de Requisitos* do CMMI e as cerimónias e artefactos do Scrum necessários para satisfazer as respetivas práticas. No capítulo 6 apresenta-se a estruturação do processo de requisitos e criada da documentação auxiliar necessária para suportar o processo definido. Desta forma foi possível satisfazer o terceiro objetivo definido para este trabalho.

O objetivo 4 foi satisfeito uma vez que foi criada documentação para suportar o processo de requisitos, tal como *templates* para a especificação de *Epics* e de *User Stories* como pode ser consultado nos anexos Anexo C e Anexo D, respetivamente. Além disso foi também criada e documentada a primeira versão do PDS da Crossing Answers que pode ser consultado no Anexo B. De forma a suportar o PDS foram criados outros anexos, que apesar de não serem referenciados neste trabalho, foram criados e documentados no âmbito da definição do PDS. São eles o Anexo E que apresenta esclarecimentos relativos a todo o ciclo do Scrum e respetivas cerimónias, artefactos e papeis, o Anexo F que apresenta

esclarecimentos relativos ao ciclo de vida do Kanban e por fim o Anexo G que apresenta a estrutura necessária para a inserção de bugs na plataforma de gestão de projeto.

Por fim no objetivo 5 era pretendido realizar a avaliação do processo após a estruturação e, no caso da existência de melhorias, proceder à atualização do mesmo e formação aos colaboradores. Para tal, no capítulo 7 foi feita a aplicação do estudo realizado num projeto da organização. Após o término do projeto foi realizada a analise comparativa com outros projetos realizados anteriormente à estruturação do processo e retiradas as principais conclusões. Foi concluído que a estruturação do processo de requisitos se verificou numa mais-valia para a organização, tendo o mesmo sido adotado pela organização. Sendo assim no capítulo 8 foi precedido à atualização do processo, tendo sido disponibilizado para todos os colaboradores da organização a documentação criada neste trabalho. Para concluir com sucesso o objetivo 5 foi necessário proceder a formação dos colaboradores, para tal foram realizados dois *workshops* com atividades práticas. Os workshops permitiram formar os colaboradores para o uso do Scrum e do Kanban e também para o processo de requisitos estruturado neste trabalho.

Concluindo, todos os objetivos deste trabalho foram concluídos com sucesso. Apesar das dificuldades existentes foi possível realizar uma base em termos de processo que irá permitir outras otimizações no futuro com maior facilidade. Isto porque já é conhecido um modelo e quais os passos necessários para usufruir do mesmo. Os *workshops* foram uma iniciativa vista com muito bons olhos tanto por parte da gerência como por parte dos colaboradores, as atividades permitiram colocar pessoas de diferentes departamentos a trabalhar em conjunto tendo sido partilhada experiência e conhecimento.

## 9.3 Principais Contribuições

Através da realização deste trabalho foi possível alcançar os vários objetivos definidos. Desta forma também foi possível fornecer várias contribuições à organização. As principais contribuições do trabalho, são:

- Um formato de otimização de processos utilizando as práticas do CMMI em abordagens ágeis (capítulo 5).
- Estruturação e otimização dos processos de desenvolvimento e gestão de requisitos e documentação da primeira versão do PDS da organização (capítulo 5 e 6).
- Definição de método para avaliação das alterações realizadas aos processos da organização (capítulo 7).
- Formação aos colaboradores da organização de forma a conseguirem usar o processo criado neste trabalho (capítulo 8).

## 9.4 Trabalho Futuro

O PDS criado neste trabalho foi, até ao momento, validado apenas num único projeto. Não obstante os resultados terem sido positivos, é desejável que sejam feitos esforços adicionais de validação, o que irá

acontecer através da adoção deste processo noutros projetos. Sendo assim, os próximos passos são antes de mais a garantia de que todo o departamento de TI se encontra a utilizar os processos definidos neste trabalho, tanto o PDS como em específico o processo de requisitos. Antes de avançar para uma nova otimização ou melhoria é necessário garantir a institucionalização da estruturação realizada neste trabalho, isto porque o conceito de institucionalização implica que o processo está enraizado na maneira como o trabalho é realizado e há compromisso e consistência na sua aplicação. Além disso segundo (CMMI Product Team, 2010; Dalton et al., 2016) um processo institucionalizado tem maior probabilidade de ser mantido durante períodos de *stress*. Alguns aspetos da institucionalização já foram considerados nas práticas genéricas apresentadas no capítulo 8.

O foco deste trabalho foi o processo de Gestão e Desenvolvimento de Requisitos. No entanto, na análise inicial foram identificadas outras áreas problemáticas, como os testes ou comunicação com o cliente. Será necessário procurar melhorias que permitam mitigar estes problemas.

# Referências


Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and analysis. *Espoo, Finland: Technical Research Centre of Finland, VTT Publications*, 478. https://doi.org/10.1076/csed.12.3.167.8613

Ahrend, J.-M. (2013). Requirements Elicitation in Startup Companies. *Research Topics in HCI*, 9. Retrieved from http://www.cs.bham.ac.uk/~rjh/courses/ResearchTopicsInHCI/2012-13/Papers/Jan.pdf

Basili, V. R., Caldiera, G., & Rombach, H. D. (2002). Goal Question Metric (GQM) Approach. *Encyclopedia of Software Engineering*, *2*, 1–10. https://doi.org/10.1002/0471028959.sof142

Beck, K. (2001). Manifesto for Agile Software Development. Retrieved from http://agilemanifesto.org/

CMMI Product Team. (2010). *CMMI® for Development, Version 1.3*. https://doi.org/CMU/SEI-2010-TR-034

Cohn, M. (2014). Getting Agile with Scrum, (January 1986).

Crossing - Digital development studio. (n.d.). Retrieved December 10, 2018, from http://crossinganswers.com/

Dalton, J., Timmerman, R., Adkins, L., Botula, K., Potter, N., Torrens, D., … Glover, M. T. (2016). *A Guide to Scrum and CMMI: Improving Agile Performance with CMMI*. Retrieved from http://cmmiinstitute.com/sites/default/files/Scrum_XP_Profile_Final.pdf

Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., & Abrahamsson, P. (2014). What do we know about software development in startups? *IEEE Software*, *31*(5), 28–32. https://doi.org/10.1109/MS.2014.129

Kniberg, H. (2011). The Multitasking Name Game.

Lina, Z., & Dan, S. (2012). Research on Combining Scrum with CMMI in Small and Medium Organizations, 554–557. https://doi.org/10.1109/ICCSEE.2012.477

Lucia, A. De, & Qusef, A. (2010). Requirements Engineering in Agile Software Development, *2*(3), 212–220. https://doi.org/10.4304/jetwi.2.3.212-220

Rules, K. P., & Mast, G. (2013). *Kanban Pizza Game Experience Kanban for yourself*.

Selleri, F., Santana, F., Soares, F., Lima, A., Monteiro, I., Azevedo, D., … Meira, D. L. (2015). Using CMMI together with agile software development : A systematic review. *INFORMATION AND SOFTWARE TECHNOLOGY*, *58*, 20–43. https://doi.org/10.1016/j.infsof.2014.09.012

What is Scrum? (n.d.). Retrieved December 10, 2018, from https://www.scrum.org/resources/what-is-scrum

# Anexo A – Levantamento da Situação Inicial

De forma geral podem ser considerados sete procedimentos que são executados pelos colaboradores dos diferentes departamentos da Crossing Answers, são eles:

1. **Proposta Comercial:** é realizada apenas em projetos para clientes e pretende fornecer como saída uma proposta comercial ou caderno de encargos validada pelo cliente.
2. **Briefing/Brainstorming:** feito em conjunto pelo COO e a gestão de projetos, tem como objetivo fazer o levantamento inicial de muito alto nível que permita criar um documento que comtemple os seguintes pontos:
    a. Funcionalidades gerais a desenvolver.
    b. Calendarizar o tempo de execução do projeto tal como as alocações necessárias a fazer.
    c. Definir ou validar as tecnologias a usar.
    d. Discutir questões de arquitetura e esboços das vistas mais importantes, se aplicável.
3. **Design e Prototipagem:** executado pela equipa de design e tem como principal objetivo produzir os layouts e *assets* necessários à equipa de desenvolvimento. Requer a validação do cliente antes de avançar para a fase seguinte.
4. **Pre-game:** no Pre-game são executadas algumas das principais tarefas que o Scrum prescreve e outras definidas internamente, são elas:
    a. Inicialização da ferramenta de gestão com um *Product Backlog* e uma *Wiki*.
    b. Definição da equipa que ira assumir o desenvolvimento.
    c. Definição de um primeiro conjunto de *User Stories* seguido de priorização e realização das respetivas estimativas recorrendo à técnica *Planning Poker*.
    d. Levantamento e definição das ferramentas a utilizar.
    e. Definição do tempo das iterações (1-4 semanas).
5. **Desenvolvimento:** tal como no ponto anterior no desenvolvimento também não foram adotadas todas as cerimônias do Scrum, sendo suposto as equipas realizarem as seguintes cerimónias: *Sprint Planning*; *Daily Meeting*; *Sprint Review* e *Sprint Retrospective*. Não existem quaisquer práticas ou artefactos que permitam realizar testes ao sistema, os que são realizados são de forma *ad hoc*. Quando o cliente pede a adição de novas funcionalidades é necessário fazer uma validação por parte da gestão de projetos deforma a avaliar o custo dessa funcionalidade. Por fim, esta fase termina quando as funcionalidades contratualizadas se encontram implementadas na sua totalidade.
6. **Aceitação:** este procedimento requer colaboração do cliente, o que nem sempre é possível. Após ser feita a apresentação final ao cliente, a operação mais comum é realizar um período de *beta testing*, em que o cliente vai reportando os problemas que encontra e são corrigidos imediatamente. Caso existam um número de alterações elevado ou novas funcionalidades é realizada uma nova iteração. Se não for suficiente ficam para uma nova versão e será feito o processo desde o início.

7. **Suporte e Manutenção:** neste procedimento não existe propriamente um protocolo definido para seguir nem uma equipa alocada exclusivamente a esta atividade. O cliente entra em contacto com a organização e internamente faz-se chegar o pedido à gestão para solucionar a questão ou problema que o cliente possa ter.

# Anexo B – Processo de Desenvolvimento de *Software*

# Processo de Desenvolvimento de Software

**CROSSING-PDS-2018-001**

**Versão 1.0**

**Desenvolvido por Filipe Rainho**

**Crossing Answers, Lda**

**2018**

# Histórico de revisões

| Nome | Data | Razão de Alterações | Versão |
|---|---|---|---|
| Filipe Rainho | 01/11/2018 | Versão inicial | 1.0 |
| | | | |
| | | | |
| | | | |
| | | | |

# Índice

# Definições e Acrônimos

A seguinte tabela apresenta os acrónimos usados ao longo deste documento.

| Acrónimo | Descrição |
|---|---|
| PDS | Processo(s) de Desenvolvimento de Software |
| PO | Product Owner |
| SM | Scrum Master |
| ST | Scrum Team |
| WIP | Work In Progress |
| | |

# Documentos Auxiliares

A seguinte tabela apresenta a lista de documentos auxiliares e documentos de referência do processo. Os documentos auxiliares e de referência são usados principalmente para fornecer mais leitura e informações complementares e/ou detalhadas.

| Referência | Documento |
|---|---|
| CROSSING-PDS-2018-002 | Scrum |
| CROSSING-PDS-2018-003 | Kanban |
| CROSSING-PDS-2018-004 | Template – Epic |
| CROSSING-PDS-2018-005 | Template – User Story |
| CROSSING-PDS-2018-006 | Template – Bug |
| | |

# 1 Introdução

Este documento tem como principal objetivo detalhar o Processo de Desenvolvimento de Software (PDS) atualmente em utilização na Crossing Answers. Fornece orientações para os colaboradores e também para os *stakeholders*, melhorando desta forma a capacidade de produzir resultados relativos ao âmbito, necessidades e expectativas dos projetos.

Atualmente, o desenvolvimento de software exige eficácia e eficiência, desta forma, os gestores e equipas de desenvolvimento precisam de entender desde o início quais os procedimentos que devem aplicar para resolver os problemas que enfrentam. Este documento descreve todas as fases do desenvolvimento e toda a organização do ciclo de vida dos projetos a desenvolver. É pretendido que este PDS seja aplicado em todos os projetos de software da Crossing Answers.

Neste documento são fornecidos os procedimentos necessários e toda a sua informação relacionada para a condução dos projetos com base neste PDS e por fim, ajudar a melhorar a eficácia e eficiência do trabalho desenvolvido pelas equipas. Sendo assim e de forma a garantir uma maior qualidade no software desenvolvido este PDS está assente nos pilares de abordagens Ágeis, sendo as frameworks contempladas o Scrum e o Kanban.

O público alvo deste documento são todos os colaborados da Crossing Answers que participam de alguma forma nos projetos de software. Além das equipas técnicas também é importante para as áreas comerciais, financeiras e de gestão da organização.

O conteúdo deste documento é propriedade da Crossing Answers, sendo que a sua disponibilização a entidades externas não é permitida.

## 1.1 Estrutura do Documento

O documento está divido em quatro secções, em seguida será feito um breve resumo do que cada uma delas irá abordar.

A primeira secção é a **Introdução**, e fornece informação sobre o documento, incluindo objetivos, propósito e também o público alvo do documento.

A segunda secção descreve fatores para a **Aplicação do Processo** nos projetos desenvolvidos. São apresentadas as diferentes configurações do processo, as suas restrições e os diferentes papeis e responsabilidades a ter em conta consoante a configuração escolhida.

A terceira secção apresenta os **Procedimentos do Ciclo de Vida do Processo**. É nesta secção que cada procedimento do processo é descrito em detalhe, tal como os seus inputs, outputs e respetivas tarefas a realizar.

A quarta secção diz respeito ás **Considerações Finais**, onde são apresentadas as boas práticas a seguir para uma melhor utilização do processo descrito neste documento.

# 2 Aplicação do Processo

O desenvolvimento de software tem um papel suficientemente importante dentro da organização para ter um ciclo de vida próprio. No desenvolvimento de software podem ser encontrados diferentes tipos de projetos, sendo uns mais exigentes do que outros em diversos fatores, tais como arquitetura, detalhe da documentação, equipa, integração de elementos externos à organização, tamanho, complexidade, tipo, entre outros.

É necessário considerar todos estes fatores de forma a encontrar a melhor maneira de gerir as características de cada caso. Em seguida são apresentados conjuntos de fatores que permitem as equipas escolher a configuração do fluxo de trabalho para o projeto, os papeis e responsabilidades a considerar para cada caso e também as restrições existentes na utilização deste PDS.

## 2.1 Tipos de Projetos

Os tipos de projeto têm com base a atividade diária da organização e o histórico de projetos já desenvolvidos, sendo possível definir classes com algumas diferenças. Cada projeto desenvolvido pela organização deve ser compatível com uma das classes existentes (apresentadas em seguida). Atualmente são utilizadas duas abordagens para suportar o desenvolvimento de software, ambas com base no manifesto ágil, são elas o Scrum (para mais detalhes sobre o Scrum consultar [CROSSING-PDS-2018-002]) e o Kanban (para mais detalhes sobre o Kanban consultar [CROSSING-PDS-2018-003]). A escolha entre elas não deve ficar ao acaso, devendo respeitar as seguintes regras.

Deve ser escolhido o Kanban caso se verifique que:

- Projetos de complexidade simples e de curta duração (inferior a um mês para um recurso);
- Módulo ou componente para integração num projeto de maior dimensão;
- Protótipos ou provas de conceito que são necessárias para projetos nos quais os requisitos não são facilmente clarificados devido à falta de conhecimento da equipa;
- Projetos de suporte e manutenção (tanto para pequenas melhorias como para correção de *bugs*).

Deve ser escolhido o Scrum case se verifique que:

- Projetos de média ou longa duração (um mês ou mais para um recurso);
- Projetos com um tempo de desenvolvimento possível de estimar;
- Projetos que irão necessitar de novos desenvolvimentos no futuro (melhorias ou novas funcionalidades);
- Projetos que representam produtos, tanto internos como para clientes;
- Serviços desenvolvidos para clientes.

Este conjunto de regras deve ser respeitado à risca. Caso exista algum cenário não previsto ou muito específico que cause dúvidas, devem ser consultados os responsáveis pelo processo antes de ser realizada a escolha.

## 2.2 Papeis e Responsabilidades

A definição dos diferentes papeis e responsabilidades do PDS permite fornecer aos colaboradores uma visão mais clara de quais são as responsabilidades de cada elemento perante um determinado projeto ou grupo de projetos. Sendo a base deste PDS assente em duas abordagens ágeis, o Scrum e o Kanban, para cada uma delas existem diferentes papeis e responsabilidades a considerar.

Nos projetos desenvolvidos com Scrum cada colaborador alocado ao projeto tem obrigatoriamente um dos seguintes cargos na equipa:

- **Product Owner (PO):** o responsável por assegurar que a equipa de desenvolvimento oferece o maior valor possível ao cliente ou ao utilizador final no menor espaço de tempo. Desta forma, o PO tem responsabilidade de alimentar e gerir o Product Backlog, garantindo também a sua priorização;
- **Scrum Master (SM):** encarregue por se certificar que os objetivos propostos para cada Sprint são realmente cumpridos conforme agendado, além disso garante a comunicação entre os diferentes papeis sendo um "escudo" para a equipa. Tem a responsabilidade de promulgar os valores e práticas do PDS;
- **Scrum Team (ST):** composta pelos elementos que trabalham em conjunto para desenvolver e entregar um incremento do sistema a cada Sprint. A ST deve ter a capacidade de autogestão.

Além dos cargos obrigatórios, os elementos da ST podem ter uma especificação ao nível das responsabilidades que vão exercer no projeto com base num conjunto de cargos não menos importantes, mas de carácter opcional neste PDS:

- **Subject Matter Experts:** são elementos da equipa especialistas em categorias de competências específicas. Nesta categoria incluem-se elementos especialistas em linguagens de programação específicas, em UX, em controlo e gestão de bases de dados, entre outros. No caso de ser identificada esta necessidade, a sua especialidade deverá ser devidamente anotada;
- **Software Quality Engineer:** elementos responsáveis por assegurar a qualidade do código produzido durante cada Sprint, criar casos de teste e desenvolver testes automatizados e manuais para testar cada um dos requisitos por completo;
- **Documentation Specialist:** este especialista é responsável por criar os manuais de utilizador, manuais de administrador, e outros documentos de treino essenciais à distribuição e comercialização do produto ou sistema desenvolvido.

Por fim, no Kanban não existe a prescrição e/ou obrigatoriedade na definição de papeis e responsabilidades. Apesar disso, segundo este PDS, é necessário que em cada projeto ou conjunto de projetos anexados a uma Kanban Board exista um responsável pelo seu controlo e gestão. Esse responsável acaba por desempenhar o mesmo conjunto de atividades do que o PO desempenha no Scrum.

## 2.3 Restrições

As restrições a serem consideradas ao iniciar um procedimento num determinado projeto são simplesmente as restrições de precedência. Neste PDS todos os procedimentos apresentados na secção seguinte devem ser aplicados sequencialmente sendo que nenhum deles deve ser ignorado.

Os critérios de entrada e os critérios de saída dos procedimentos são de caracter obrigatório por defeito, quando forem opcionais serão referenciados com tal. Relativamente ás tarefas a executar em cada um dos procedimentos, deve ser seguido o processo tendo em conta a abordagem escolhida para o projeto (Scrum ou Kanban).

# 3 Procedimentos do Ciclo de Vida do Processo

Nesta secção são apresentados os diferentes procedimentos do ciclo de vida do PDS, incluindo todos os *inputs* e *outputs* de cada procedimento e também a descrição do processo de transformação dos *inputs* de cada procedimento nos respetivos *outputs*. Cada um dos procedimentos é um indicador de estado e de tempo do projeto que estiver a ser desenvolvido segundo as regras do PDS. Todos os procedimentos devem ser executados sequencialmente e não devem ser ignorados ou colocados de parte. Na Figura 1 é apresentado um diagrama com os procedimentos ordenados sequencialmente conforme previsto neste PDS.


*Figura 1 - Procedimentos do ciclo de vida do PDS*

Todos os projetos devem ter início no procedimento "Pre-game" e posteriormente seguir os restantes pela ordem apresentada. Em cada procedimento existem regras definidas que devem ser respeitadas ao máximo. É essencial que a ordem dos procedimentos seja respeitada pelas equipas de forma a criar os objetos de trabalho definidos no PDS correta e atempadamente.

Os procedimentos são organizados num ciclo de vida que utiliza processos para produzir os objetos de trabalho relevantes e necessários para a execução e conceção dos projetos. Nesta secção são descritos em maior detalhe cada um dos procedimentos do PDS.

## 3.1 Pre-game

O primeiro procedimento do PDS é denominado de "Pre-game". Este procedimento divide-se em duas grandes fases, são elas a fase de planeamento e a fase de definição de arquitetura de alto nível.

A fase de planeamento tem como tarefa principal a definição do Product Backlog, que representa a lista de funcionalidades a serem desenvolvidas de acordo com as necessidades do cliente ou dos *stakeholders*. Para tal ser possível é necessário passar por uma fase de elicitação de requisitos, de forma a definir as funcionalidades principais com base no que o cliente e/ou *stakeholders* procuram e por sua vez definir um Product Backlog. Além da definição do Product Backlog existem outras tarefas ou pontos a discutir na fase de planeamento, tais como:

- Preparação da ferramenta de gestão de projeto;
- Priorização do Product Backlog;
- Estimação do Product Backlog;
- Calendarização da entrega;
- Definição da estrutura da equipa;
- Ferramentas necessárias.

A fase de planeamento no “Pre-game” pode ser muita ampla caso se trate de um novo sistema, mas caso seja uma nova versão ou apenas melhoria a um sistema já existente o seu tempo poderá ser mais reduzido. Por fim, após a fase de planeamento é realizada a fase de definição da arquitetura de alto nível do sistema a desenvolver, com exceção de projetos de suporte e manutenção aos quais esta fase não se aplica.

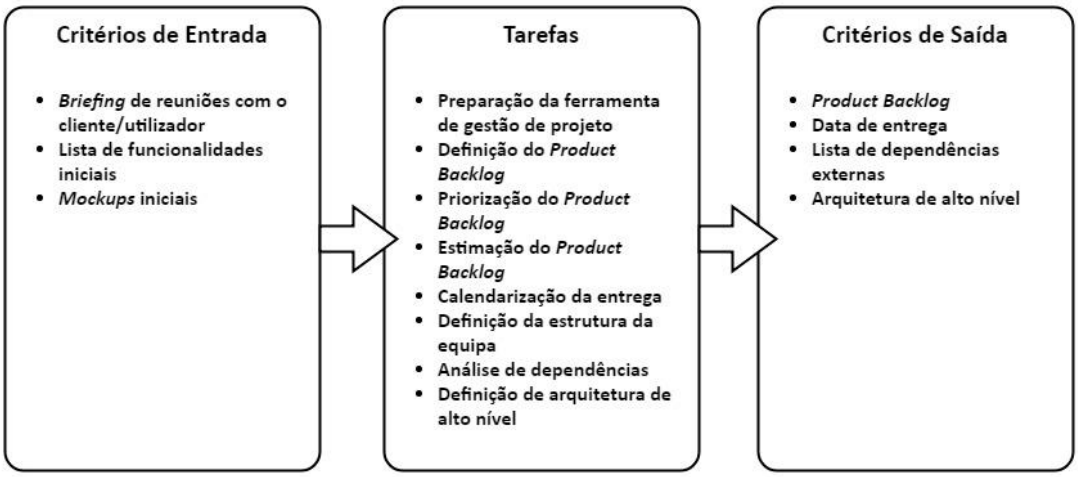



*Figura 2 - Procedimento "Pre-game"*

### 3.1.1 Critérios de Entrada

Os critérios de entrada do “Pre-game” são:

- O *briefing* resultante das reuniões com o cliente/utilizador;
- A lista de funcionalidades iniciais;
- Os *mockups* desenvolvidos pela equipa de design.

Através deste conjunto de informação a equipa dispõe de uma base aceitável para concluir com sucesso as tarefas enumeradas na lista deste procedimento.

### 3.1.2 Tarefas

Este procedimento pode tornar-se algo complexo, especialmente se for um sistema novo. Além disso, e não menos importante, a complexidade, o número de riscos e a inexistência de histórico são outros fatores que aumentam a complexidade e consequentemente o tempo de execução. Em primeiro lugar deve ser realizada a Preparação da ferramenta de gestão de projeto e em seguida as seguintes tarefas:

- Definição do Product Backlog;
- Estimação do Product Backlog;
- Priorização do Product Backlog;
- Calendarização da entrega;
- Definição da estrutura da equipa;
- Análise de dependências;
- Definição de arquitetura de alto nível do sistema.

Em seguida são descritas cada uma das tarefas em mais detalhe, sendo fornecidas todas as fases, objetivos e diretivas para cumprir e respeitar o PDS.

#### 3.1.2.1 Preparação da ferramenta de gestão de projeto

A escolha da abordagem a utilizar no desenvolvimento deve ser feita com base nos critérios enumerado em Tipos de Projetos. As regras aí estabelecidas devem ser cumpridas minuciosamente.

Tanto no Scrum como no Kanban deve ser usada uma Collection onde a equipa pode consultar o estado do *work items* e visualizar o *workflow* facilmente, permitindo assim ter visivel toda a cadeia de valor e os *work items* no contexto correto. A Crossing Answers prevê o uso do Favro para realizar a gestão de projetos.

O Favro é uma ferramenta de gestão e controlo de *workflow* de projetos, que aliando a flexibilidade à sua componente visual permite aos colaboradores gerir todos os aspetos relacionados com a atividade operacional de forma fácil e intuitiva. Através de um simples *drag and drop* é possível alterar o estado dos *work items*, permitindo aos restantes elementos estar a par do desenvolvimento do trabalho de cada colega. Para além disso o Favro oferece outras funcionalidades úteis para auxiliar o controlo do desenvolvimento com base em Scrum ou em Kanban.

O uso da ferramenta é obrigatório em todos os projetos desenvolvidos na empresa, seja eles projetos internos ou para clientes. Sempre que é iniciado um novo projeto deve ser criada uma nova Collection. Existem dois *templates* base de Collections criados na ferramenta, um deles para projetos a ser desenvolvidos com base em Scrum e outro para projetos a serem desenvolvidos com base em Kanban. Em cada projeto a responsabilidade da criação, da configuração e da gestão da Collection é do PO. Para tal, existem *templates* disponibilizados na ferramenta, tanto para a criação como para as configurações necessárias.

#### 3.1.2.2 Definição do Product Backlog

A definição do Product Backlog prevê a definição de Epics e de User Stories. Para proceder à especificação, quer seja nesta fase ou mais tarde após o desenvolvimento ter iniciado, devem ser utilizados com base os critérios de entrada deste procedimento. Além disso pode ser necessário obter mais clareza em alguma da informação fornecida. Uma vez que as necessidades do cliente e do negócio podem mudar devem ser usadas técnicas de elicitação de requisitos, tais como entrevistas, *brainstorming*, prototipagem ou simplesmente uma reunião para recolha de *feedback*. A abordagem a utilizar pode variar de projeto para projeto, sendo necessário, antes de mais, entender a natureza do mesmo. O *brainstorming* é aconselhável quando as ideias não são bem claras e ainda existem muitas dúvidas em relação ao funcionamento do sistema. A prototipagem é uma técnica bastante eficaz, sendo a sua aplicação aconselhada quando o fator tempo não é o mais relevante e quando o cliente pretende bastante rigor no sistema. Por fim as entrevistas são bastante úteis quando se lida com um cliente, já que permitem perceber mais facilmente o que o mesmo pretende, evitando assim desperdícios de tempo e recursos na definição do Product Backlog. Estas técnicas podem ser usadas tanto nesta fase como posteriormente no Backlog Grooming.

Após o PO considerar que tem informação suficiente deve proceder à definição do Product Backlog começado pela especificação das Epics iniciais. O processo de desenvolvimento passa por reunir os elementos ligados ao projeto para discutir as Epics mais apropriadas para o

projeto. Todas as Epics devem ser definidas com a maior clareza possível tendo em conta os recursos existentes no início do projeto. O número de Epics pode crescer com o decorrer do desenvolvimento do projeto, tal como as Epics existentes podem necessitar de ser aprimoradas. Tipicamente, a definição das Epics é uma tarefa da responsabilidade do PO, no entanto pode ser requisitado apoio.

Após a definição das Epics iniciais a ST deve proceder à validação das Epics desenvolvidas, garantindo que são claras o suficiente para proceder ao processo de refinar as Epics em User Stories. Caso a ST considere que existem Epics pouco claras (diferentes projetos podem ter diferentes níveis de clareza), mesmo para uma fase inicial, o PO deve complementar a especificação, mesmo que para isso necessite de recorrer novamente a técnicas de elicitação.

Concluído o processo de validação, segue-se o processo de refinar as Epics em User Stories. Este processo deve ser feito na presença de toda a equipa e considerando todo o material existente e o que foi definido até ao memento. O Product Backlog não necessita de ser definindo na totalidade, pode ser feito, continuamente, à medida que os *stakeholders* vão especificando melhor as suas necessidades e também com a recolha de *feedback* nas demonstrações que serão realizadas. Para tal podem ser utilizadas todas as técnicas de elicitação referidas anteriormente.

Independentemente do conjunto de técnicas usadas, o seu resultado deve produzir sempre um conjunto de Epics e de User Stories de forma a definir o Product Backlog do projeto em questão. As Epics definidas devem seguir o formato apresentado em [CROSSING-PDS-2018-004] e as Users Stories devem seguir o formato apresentado em [CROSSING-PDS-2018-005].

#### 3.1.2.3 Estimação do Product Backlog

No caso do Scrum o Product Backlog irá ser estimado em *story points*. Os *story points* são uma medida estimada do esforço necessário para a realização de cada User Story. A atribuição a cada User Story do projeto será feita através de uma reunião com toda a equipa, onde cada elemento apresenta a sua classificação para cada User Story. Em caso de discórdia na pontuação a atribuir, haverá uma discussão para clarificar o trabalho a realizar na User Story e também a sua complexidade, de forma a tentar chegar a uma pontuação unânime, ou pelo menos maioritária.

O método utilizado é o Planning Poker, seguindo a escala Fibonacci com um limite predefinido para a pontuação. Neste método cada elemento da equipa apresenta a sua estimativa em simultâneo e, em equipa, discutem para chegar a uma estimativa final.

#### 3.1.2.4 Priorização do Product Backlog

No caso do Scrum a prioridade inicial atribuída ás User Stories não deve ser deixada ao acaso. Para tal deve ser realizada uma reunião com a equipa e o cliente ou o representante do mesmo (se possível) que deve ter como resultado o Product Backlog priorizado, da mais prioritária para a menos prioritária. É importante que quando no procedimento seguinte sejam iniciados os desenvolvimentos (seja com Scrum ou com Kanban) o Product Backlog já se encontre priorizado.

#### 3.1.2.5 Calendarização da entrega

No planeamento de qualquer projeto, independentemente da sua tipologia, deve ser definida sempre uma data de entrega. É importante definir qual será o período de tempo que a

equipa irá ter para desenvolver o conjunto de Epics e User Stories definidos à partida. A forma como a estimativa do tempo do projeto deve ser realizada depende do histórico de projetos da empresa, das características do projeto atual, dos seus riscos, e da experiência da equipa envolvida no projeto. Desta forma existem duas possibilidades:

1. A empresa tem no seu histórico outros projetos semelhantes ao projeto que se pretende estimar, devendo basear-se nesse histórico para obter uma estimativa mais precisa para o projeto atual;
2. A empresa não possui esse histórico, sendo a estimativa inicial do tempo do projeto baseada na estimativa de cada um dos elementos da equipa.

#### 3.1.2.6 Definição da estrutura da equipa

Consoante a abordagem escolhida (Scrum ou Kanban) deve ser definida a estrutura da equipa. Para consultar em mais detalhe os papeis e responsabilidade existentes deve ser consultado o conteúdo de Papeis e Responsabilidades. Apesar do Scrum prescrever equipas com uma dimensão entre 5-9 elementos, no caso deste PDS as equipas podem ter qualquer número de elementos até um máximo de 9. Existem pequenos projetos que por vezes se resumem a equipas de 2-3 pessoas devido à baixa complexidade e ao baixo número de funcionalidades requeridas. A atual dimensão da empresa também é um fator que influencia diretamente o tamanho das equipas.

#### 3.1.2.7 Análise de dependências

Com base nas especificações do Product Backlog, é necessário identificar as dependências externas do projeto e as ferramentas necessárias. Esta identificação assegura que o progresso das atividades não é interrompido e que as várias dependências são resolvidas com a antecedência necessária.

Por dependências externas entendem-se todos os componentes que são da responsabilidade de terceiros, não integrantes do projeto, ou quaisquer recursos externos que necessitem de ser assegurados antes da realização de alguma etapa, por exemplo produtos ou serviços de terceiros, dependências técnicas (hardware, servidores, etc.), dependências legais e dependências internas à organização (trabalho a ser realizado por outra equipa ou secção).

No caso de serem identificadas dependências externas, as mesmas devem ser registadas na secção apropriada na ferramenta de gestão de projeto de forma a ser visíveis a qualquer altura para a equipa, tal como o seu estado de disponibilização.

#### 3.1.2.8 Definição de arquitetura de alto nível do sistema

Ainda neste procedimento, a equipa deverá decidir, em conjunto, a arquitetura de alto nível do sistema. Neste PDS a arquitetura não é, no entanto, um elemento rígido uma vez que poderá ser alterada na totalidade se as necessidades mudarem ou se a escolha inicial se revelar ineficiente.

Do mesmo modo, a decisão inicial sobre a estrutura da arquitetura não precisa de ser completa. Deve incluir os elementos mais importantes de estrutura e comportamento do produto, mas não precisa de ser um desenho arquitetural extremamente detalhado e pormenorizado. Desta forma esta tarefa pode ser dividida em três principais etapas:

1. Identificação dos requisitos não-funcionais relevantes para a arquitetura;

2. Identificação dos principais protocolos e padrões a utilizar;
3. Elaboração pequenos diagramas, esquemas e documentos de arquitetura de alto nível do sistema.

Esta tarefa não se aplica a projetos de suporte e manutenção, uma vez que não serão desenvolvidas novas funcionalidades nem feitos ajustes à arquitetura existente.

### 3.1.3 Critérios de Saída

Assim que se encontrem concluídas todas as tarefas o procedimento "Pre-game" chega ao fim. Desta forma, a equipa deve conseguir dar resposta aos seguintes critérios de saída:

- O Product Backlog do projeto a desenvolver ordenado por prioridade e estimado pelo menos para o que a equipa prevê como o primeiro Sprint;
- A data de entrega definida (com exceção de protótipo e projetos com requisitos para os quais não foi possível realizar uma estimativa por parte da equipa);
- A lista de dependências externas (caso existam);
- Toda a documentação e decisões tomadas pela equipa em questões da arquitetura de alto nível (caso existam).

## 3.2 Desenvolvimento

O segundo procedimento do PDS é denominado de "Desenvolvimento". Este procedimento tem como principal foco o desenvolvimento propriamente dito do sistema. Com base no presente documento foram apresentadas duas hipóteses para gerir o *workflow* deste procedimento, foram elas o Scrum e o Kanban. Os responsáveis fizeram a escolha com base nas restrições fornecidas em Tipos de Projetos.

Em seguida são apresentadas, em detalhe, um conjunto de tarefas adjacentes ao procedimento "Desenvolvimento", tal como os critérios de entrada necessários e também os critérios de saída que devem ser produzidos.

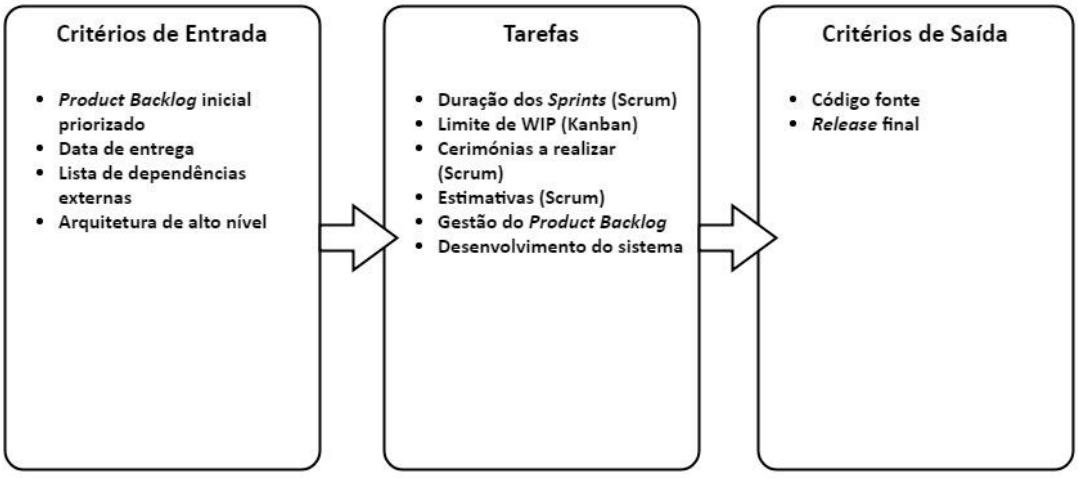



*Figura 3 - Procedimento "Desenvolvimento"*

### 3.2.1 Critérios de Entrada

Os critérios de entrada do "Desenvolvimento" são:

- O Product Backlog inicial priorizado (na ferramenta de gestão de projeto);
- A data de entrega definida;
- A lista de dependências externas (opcional);
- Documentação relacionada com questões de arquitetura de alto nível (opcional).

Além da informação definida como critérios de entrada, caso seja necessário, a equipa deve ter acesso a toda a documentação produzida no procedimento anterior, tal como toda a informação extra conveniente ao projeto.

### 3.2.2 Tarefas

Algumas das tarefas deste procedimento só se aplicam a uma das abordagens (ou Scrum ou Kanban), desta forma a equipa deve considerar apenas as que dizem respeito para o projeto em questão. As tarefas a executar neste procedimento são descritas detalhadamente em seguida, sendo fornecidos os objetivos e as diretivas a respeitar de forma a cumprir o PDS.

#### 3.2.2.1 Duração dos Sprints (Scrum)

Uma das primeiras decisões que a equipa deve tomar é a duração que cada Sprint irá ter, tendo em conta o que faz mais sentido para a equipa e para o projeto. Como já referido anteriormente a duração definida inicialmente pode ser posteriormente alterada, mas só entre o final e o início do Sprint seguinte. Contudo não é recomendável fazer alterações à duração de forma frequentemente, já que irá dificultar o trabalho da equipa no planeamento do Sprint seguinte.

Os Sprints podem variar entre uma a quatro semanas, dependendo das características do projeto e da frequência de entrega de *deliverables* pretendida pelo cliente. O principal fator para a decisão do tempo de cada Sprint é a frequência que a equipa necessita de *feedback* do cliente tal como a necessidade que o cliente tem em validar o que é desenvolvido e também a velocidade com que os requisitos do negócio podem tender em mudar.

Nos projetos desenvolvidos na Crossing Answers é aconselhado optar por 1-2 semanas de duração para cada Sprint.

#### 3.2.2.2 Limite de WIP (Kanban)

Limitar a quantidade de WIP ajuda as equipas a não criarem compromisso com muito trabalho em simultâneo. O WIP é limitado por cada estado definido na Kanban Board, sendo permitidas alocações de novas funcionalidades até que não seja excedido o limite para o estado em questão. O limite do WIP para cada estado é personalizado para cada projeto e a sua definição deve ter em conta vários fatores, nomeadamente a dimensão da equipa, a complexidade do projeto e a granularidade com que as User Stories são definidas.

De forma a dar resposta eficientemente ás necessidades mais inesperadas e urgentes do cliente a qualquer altura, deve existir disponível em todas as Kanban Board uma *priority lane* (ou *fast lane*). O seu objetivo principal é a resolução de problemas críticos e inesperados para tal percorre todos os estados da Kanban Board tendo um limite próprio para o WIP, sendo o recomendado um máximo de um *work item* por cada elemento da equipa.

#### 3.2.2.3 Cerimónias a realizar (Scrum)

De modo a assegurar o correto funcionamento do Scrum são definidas algumas cerimónias a realizar pela a equipa de desenvolvimento. A equipa terá de cumprir rigorosamente as cerimónias pois estão diretamente relacionadas com o sucesso do projeto, uma vez que fornecem transparência e clareza a todos os envolvidos no projeto.

- **Sprint Planning:** Reunião realizada no início de cada Sprint ente a equipa e o PO de forma a negociar qual o valor a entregar no respetivo Sprint. O PO apresenta a visão de trabalho para a iteração e a equipa discute a quantidade de trabalho que consegue realizar durante o Sprint. O resultado da reunião deve ser o Sprint Backlog e o compromisso da ST na sua implementação.
- **Daily Scrum Meeting:** É uma breve reunião realizada diariamente com a duração máxima de 15 minutos. Todos os elementos da equipa de desenvolvimento devem participar, sendo o objetivo da mesma fazer um ponto de situação do trabalho realizado e a realizar. Para isso devem ser respondidas três perguntas:
    - O que fiz ontem?
    - O que vou fazer hoje?
    - Foram identificados problemas que estão no caminho?
- **Backlog Grooming:** Reunião realizada de forma a produzir um Product Backlog priorizado e atualizado ao longo do desenvolvimento, podendo o memso ser realizado a qualquer altura. O Backlog Grooming inclui negociação entre o PO e a ST sobre as Epics e User Stories que serão adicionadas, removidas, atualizadas ou refinadas. Todos os *stakeholders* relevantes podem contribuir nas decisões a ser tomadas, tornando o Backlog Grooming uma atividade crítica para o planeamento e desenvolvimento dos requisitos do projeto.
- **Sprint Review:** Reunião realizada no final de cada Sprint e imediatamente antes da Sprint Retrospective. Existe um responsável da equipa que apresenta o trabalho produzido aos *stakeholders* de forma a que fiquem cientes do que está a ser entregue no final de cada Sprint. Deve ser confrontado o trabalho que foi realizado com aquele que foi planeado para o Sprint. Pode ser considerada uma forma de comunicação, validação e reconhecimento. Por fim, pode ainda servir como forma dos *stakeholders* acrescentarem ou alterarem requisitos existentes no Product Backlog.
- **Sprint Retrospective:** Reunião realizada no final de cada Sprint, de forma a discutir como correu e quais as melhorias a aplicar nos processos de forma a que os erros praticados no passado não sejam repetidos. Como preparação da reunião, que tem um máximo de 30 minutos, os elementos da equipa devem apontar os aspetos positivos, negativos e melhorias para que possam ser discutidos. A discussão deve ter como resultado a resposta as seguintes perguntas:
    - O que fizemos bem?
    - O que podíamos ter feito melhor?

#### 3.2.2.4 Estimativas (Scrum)

As estimativas são uma parte crucial do desenvolvimento quando se opta por uma abordagem Scrum. No procedimento “Pre-game” já foi apresentada a técnica e a escala a utilizar nas estimativas das User Stories.

Para estimar o trabalho a alocar a cada Sprint, a equipa deve, idealmente, ter a noção da sua velocidade (*story points*/Sprint). De seguida, a equipa deve analisar as User Stories no Product Backlog e selecionar para o Sprint um conjunto de User Stories de maior prioridade. A soma dos *story points* desse conjunto de User Stories deve ser o mais próximo possível da velocidade atual da equipa, permitindo à equipa garantir um compromisso.

A equipa deve manter o seu progresso para cada Sprint num *burndown chart*. Este *burndown chart* deve ser descritivo dos *story points* a executar e dos já foram concluídos. O objetivo será criar um registo visual do trabalho já feito e do trabalho que ainda falta realizar.

#### 3.2.2.5 Gestão do Product Backlog

No procedimento "Pre-game" existe uma tarefa associada a preparação da ferramenta de gestão de projeto, é também nessa ferramenta que deve ser feita toda a gestão do Product Backlog. Sendo o Product Backlog propriedade do PO é da sua responsabilidade a manutenção e gestão. Existem várias operações associadas à gestão do Product Backlog, tais como a adição, a remoção, a atualização, estimar e ajustar estimativas existentes e por fim a sua priorização. Para proceder a realização de qualquer uma destas operações deve ser realizada uma reunião de Backlog Grooming, onde são realizadas várias negociações entre o PO e a ST. O processo de negociação deve incluir uma validação das especificações fornecidas pelo PO por parte da ST, garantindo que existe clareza suficiente na especificação fornecida.

De forma a manter a clareza, coerência e formato das Epics e das User Stories deve ser respeitada um conjunto de aspetos importantes na definição dos requisitos. Para tal foram produzidos documentos com o formato pretendido e com exemplos para cada formato, podem ser consultados em [CROSSING-PDS-2018-004] e [CROSSING-PDS-2018-005]. Ainda relativamente à questão da clareza, o PO e a ST são livres de recorrer a técnicas de elicitação sempre que necessário.

Apesar de algumas das práticas apresentadas serem prescritas apenas no Scrum, mesmo nos projetos geridos com o Kanban pode ser usado o mesmo ciclo para a gestão do Product Backlog.

#### 3.2.2.6 Desenvolvimento do sistema

O processo de desenvolvimento do produto em si pode seguir o processo do Scrum ou o do Kanban. Durante o desenvolvimento é criado o código fonte de acordo com o Product Backlog. O Product Backlog pode ser dividido em vários Sprints no caso do Scrum e integrado no produto final no final de cada Sprint ou então desenvolvido continuamente no caso do Kanban.

No desenvolvimento do sistema o processo de validação, independentemente da abordagem escolhida, é recolher regularmente *feedback* do cliente em relação aos *deliverables* produzidos. O *feedback* produzido deve ser devidamente anotado no registo de reunião criado. Além disso, em sistemas ou funcionalidades mais críticas, pode ser feita revisão de código. As revisões devem produzir uma entrada na ferramenta de gestão e as ilações retiradas devem ser devidamente anotadas e, se necessário, proceder às correções dos problemas ou más práticas de programação encontradas.

### 3.2.3 Critérios de Saída

Assim que todas as tarefas se encontrem concluídas e exista uma *release* final para entregar ao cliente, o procedimento "Desenvolvimento" chega ao fim. Os critérios de saída que a equipa deve fornecer são:

- Código fonte: deve ser fornecido de forma a ser armazenado e a ser criado um registo no histórico da organização;
- *Release* final: a demo aprovada pelo cliente no último Sprint. É esta versão que será colocada em produção no procedimento seguinte.

## 3.3 Post-game

O terceiro procedimento do PDS é o "Post-game". O objetivo desta etapa pós-desenvolvimento passa por concluir e entregar a *release* produzida pela equipa e planear e executar a disponibilização do sistema desenvolvido.

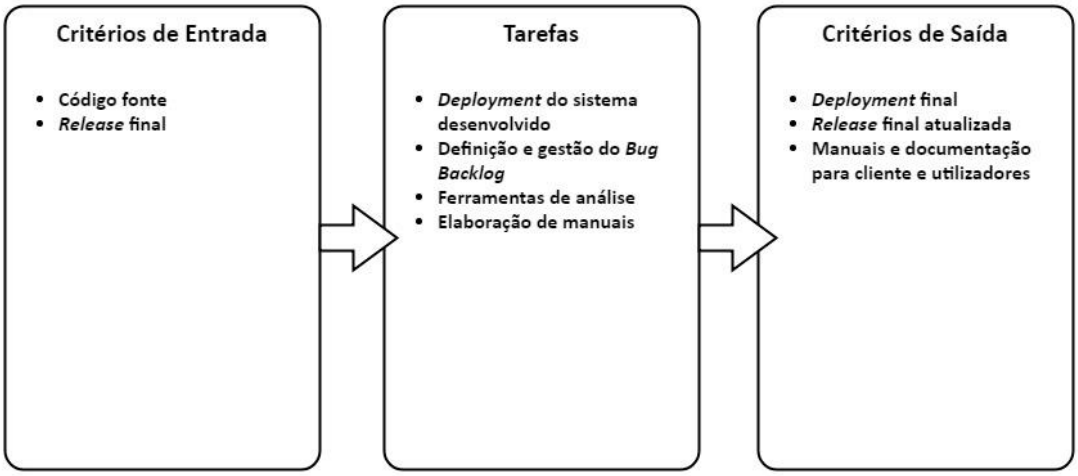



*Figura 4 - Procedimento "Post-game"*

### 3.3.1 Critérios de Entrada

Os critérios de entrada definidos para o procedimento "Post-game" são o resultado de algumas das tarefas executadas no procedimento anterior. Os critérios de entrada são:

- Código fonte do projeto;
- *Release* final – última *demo* aprovada.

Além dos elementos definidos como critérios de entrada neste procedimento, caso seja necessário, a equipa deve ter acesso a toda a documentação produzida nos procedimentos anteriores.

### 3.3.2 Tarefas

Apesar de o projeto desenvolvido já ter uma *release* final, as tarefas a executar neste procedimento não deixam de ser importantes. As equipas devem conseguir gerir todos os

aspetos posteriores ao desenvolvimento do projeto. As tarefas a executar neste procedimento detalhadas em seguida.

#### 3.3.2.1 *Deployment* do sistema desenvolvido

O *deployment* é uma tarefa muito importante do processo de desenvolvimento de um software, isto porque é o momento em que o sistema desenvolvido passa a estar em produção para os utilizadores finais. Atualmente não existem processos de automatização do processo de *deployment* nem mecanismos de entrega contínua definidos na Crossing Answers. Todo o processo é feito manualmente pela equipa de desenvolvimento ou pelo responsável de DevOps do projeto (caso se justifique delegar um colaborador).

Para facilitar este processo, ainda que seja manual, aquando da criação do repositório de código devem ser definidas, sempre, três *branches*, são elas: "production", "acceptance" e "dev". O *deployment* deve ser sempre feito com o código presente na *branch* "production". No caso das *demos* para apresentação e aceitação do cliente, o código fonte utilizado será o código presenta na *branch* "acceptance". É possível concluir que todo o processo de *deployment* realizado atualmente tem como base os repositórios de código fonte, este aspeto deve ser considerado e respeitado por todas as equipas de forma a garantir um maior controlo das versões que são disponibilizadas aos clientes e também a normalização entre os diferentes projetos desenvolvidos.

#### 3.3.2.2 Definição e gestão do Bug Backlog

Nos *templates* criados na ferramenta de gestão de projeto existe um Bug Backlog, que tem como objetivo permitir o armazenamento e gestão dos *bugs* reportados de forma categorizada e priorizada (do mais prioritário para o menos prioritário). A prioridade deve ser discutida entre a equipa e o cliente, sendo nesse momento também feita a classificação do *bug* em questões de severidade. Existem quatro categorias de severidade para a classificação de *bugs*, são elas:

- **Minor:** problemas simples que não ocorrem sempre e quando ocorrem não impedem o funcionamento do sistema ou então questões relativas ao layout de uma determinada vista ou conjunto de vistas (por exemplo: alinhamento de componentes, animações, etc.).
- **Normal:** problema que impede a utilização de uma funcionalidade do sistema, mas apenas num cenário específico (por exemplo: falta de validação ou validação errada de um campo, a pesquisa por um determinado campo não está a funcionar, etc.).
- **Important:** problema que impede a utilização de uma funcionalidade do sistema, mas não o sistema na totalidade (por exemplo: um formulário de inserção na inserção o registo não tem qualquer ação ou apresenta sempre erro, a funcionalidade de pesquisa numa lista não funcionar de todo, etc.).
- **Critical:** problema que impede a utilização de todo o sistema, ou seja, mesmo não existindo problemas noutras funcionalidades o problema em questão afeta todo o sistema (por exemplo: não ser possível carregar o sistema, ou o sistema não passar da autenticação, etc.) ou então a funcionalidade afetada é tão necessária na ótica do negocio que não faz sentido usar o sistema sem ela (por exemplo: um sistema de faturação não deixar emitir faturas, ou um sistema de chat não permitir enviar/receber mensagens).

O Bug Backlog está presente na Collection do projeto desde o início do projeto, mas só devem ser registados *bugs* encontrados em produção (pós *deployment*), caso contrário são reportados diretamente à equipa de desenvolvimento. No caso de ser reportado diretamente à equipa de desenvolvimento, se estiver ligado a um requisito existente deve ser associado ao mesmo e verificar a necessidade de voltar a estimar. Caso não tenha ligação a um requisito existente passa a ser categorizado como um novo requisito que deve ser documentado, estimado e alocado para desenvolvimento na altura certa.

Além das características enumeradas anteriormente, existem outras a ser consideradas quando um *bug* é registado no Bug Backlog, para tal podem ser consultados exemplos e guias em [CROSSING-PDS-2018-006].

#### 3.3.2.3 Ferramentas de análise

Deverá ser usado um sistema que permita fazer o *tracking* do estado da aplicação, identificando o número de *crashes* que estão a acontecer, o número de utilizadores afetados, a versão do sistema que estão a utilizar, entre outras métricas que a equipa ache uteis para a sua análise.

Desta forma a equipa tem uma ferramenta que identifica de forma clara os *crashes* mais frequentes que afetam um maior número de utilizadores, podendo assim debruçar-se sobre os incidentes que considerem mais relevantes.

Neste momento a Crossing Answers só tem prescrição para o caso das aplicações móveis, sendo que a escolha é o *Fabric.io*.

#### 3.3.2.4 Elaboração de manuais

É também neste procedimento que é feita a elaboração de toda a documentação que será necessária para fornecer ao cliente e aos utilizadores finais do sistema desenvolvido. A documentação elaborada pode ter vários formatos e propósitos. Pode variar entre manuais de utilizador em formato de impressão, ou conteúdo para páginas Web informativas do sistema desenvolvido, entre outros tópicos pertinentes para o sistema em questão.

### 3.3.3 Critérios de Saída

Assim que todas as tarefas se encontrem concluídas e exista uma nova *release* final para entregar ao cliente o procedimento “Post-game” chega ao fim. Os critérios de saída são:

- *Deployment* da última versão de código produzido na *release*.
- Atualização da r*elease* final com os problemas encontrados já resolvidos;
- Manuais e documentação para o cliente e utilizadores finais.

# 4 Considerações Finais

Este PDS foi construído de modo a poder ser facilmente aplicado a qualquer projeto de software da Crossing Answers. A sua utilização não deve dispensar a consulta dos anexos que acompanham este documento. Os anexos podem conter informação detalhada que ajude a equipa num caso de um projeto mais específico.

Este PDS está sujeito a uma evolução e melhoria constante através da sua utilização em projetos desenvolvidos na Crossing Answers ou por sugestões dadas pelos colaboradores. O objetivo é que o processo seja cada vez mais prático e eficaz, e que sejam corrigidas as lacunas encontradas perante a sua utilização nos diversos projetos. Desta forma as equipas devem garantir que utilizam a versão mais atualizada do PDS.

# Anexo C – *Template Epic*

# Template - Epic

**CROSSING-PDS-2018-004**

**Versão 1.0**

**Desenvolvido por Filipe Rainho**

**Crossing Answers, Lda**

**2018**

# Histórico de revisões

| Nome | Data | Razão de Alterações | Versão |
|---|---|---|---|
| Filipe Rainho | 01/10/2018 | Versão inicial | 1.0 |
| | | | |
| | | | |
| | | | |
| | | | |

# Definição de Epics

A incerteza sobre os requisitos, especialmente no início do projeto, impulsiona a necessidade de documentar o que é conhecido num alto nível, levando à criação de Epics. Em termos ágeis, uma Epic é uma grande User Story que irá ser necessário refinar em User Stories mais pequenas antes de serem alocadas a um Sprint. Esta escolha é feita pelas equipas de forma a facilitar a gestão dos requisitos e ser possível completar as User Stories num único Sprint.

As Epics são um *work product* criado como resultado do processo de eliciatação e desenvolvimento de requisitos. Posteriormente são refinadas em User Stories durante o Backlog Grooming, ou durante o planeamento de uma *release*, ou de forma contínua pelo Product Owner ao longo do desenvolvimento do projeto. Este processo é geralmente iterativo.

O Product Owner pode apresentar um requisito à equipa achando que se trata de uma User Story, apesar disso a equipa pode considerar que é uma Epic e fazer uma maior divisão. Não existe uma medida para diferenciar uma Epic de uma User Story, embora seja comum considerar que uma User Story pode abranger vários Sprints, mas uma Epic não. A alocação aos Sprints não é a única razão para refinar as Epics, um outro motivo é a clareza que o requisito necessita para ser desenvolvido.

Os clientes geralmente pedem funcionalidades complexas que precisam de ser refinadas em componentes menores para serem entendidos pela equipa de desenvolvimento. Por fim, e não menos importante a necessidade de alocar mais do que uma equipa para a realização do trabalho, sendo que nestes casos é, automaticamente, necessária uma divisão em User Stories menores.

Na Tabela 1 é apresentada a listagem de atributos que uma Epic deve conter e também um exemplo prático para cada um dos atributos.

| # | Título | Descrição | Exemplo |
|---|---|---|---|
| 1 | ID | Identificador único para a Epic. É inserido automaticamente pelo sistema. | *Código único*. |
| 2 | Título | Cada Epic deve ter um título que, em resumo, explica a Epic. O título não deve ter mais do que uma frase. | Gestão de permissões de utilizadores e de níveis de acesso. |
| 3 | Descrição | A descrição contém informações mais detalhadas sobre a Epic. Além disso deve ser descrito o porquê e o benefício da sua implementação no sistema. | O sistema deve permitir gerir e atribuir níveis de acesso e permissões aos utilizadores do sistema por parte dos administradores.<br>PORQUÊ: Nem todos os utilizadores podem ter acesso a todo o conteúdo da plataforma.<br>BENEFÍCIO: Permite aos administradores centralizar |

| | | | |
|---|---|---|---|
| | | | toda a informação operacional na plataforma disponibilizando apenas a que quiserem aos utilizadores que pretenderem. |
| 4 | Origem | De onde surgiu a Epic? Pode ser de uma pessoa, de uma reunião ou de um documento. | Reunião de dia <XX-XX-XXXX> com o cliente. |
| 5 | Lista de User Stories associadas | Cada Epic deve conter a listagem das User Stories refinadas através da Epic em questão, isto de forma a serem apresentadas todas as suas relações, ajudando a manter a rastreabilidade entre os requisitos. | |
| 6 | Criado por | O elemento que procedeu à definição e criação da Epic no Product Backlog. É inserido automaticamente pelo sistema. | *Nome*. |
| 7 | Data de criação | Data de criação. É inserido automaticamente pelo sistema. | |
| 8 | Data de atualização | Data da última atualização. É inserido automaticamente pelo sistema. | |

*Tabela 1 - Template de uma Epic*

De forma a trazer alguma clareza ao processo de criação de Epics (e posteriormente das User Stories) podem ser usadas algumas técnicas para ajudar a trazer robustez ao Product Backlog, são elas:

- Considerar todas as fontes de informação pertinentes ao projeto e não apenas a informação disponibilizada pelo cliente. Perder requisitos pode ter um impacto negativo no futuro.
- Incluir todos os *stakeholders* relevantes no processo de criação e revisão das Epics de forma a mitigar possíveis mal-entendidos e evitar refazer trabalho nos Sprints fututos.
- Estabelecer que a rastreabilidade é algo necessário na gestão de requisitos, independentemente da forma como é feita (neste caso através da associação).

# Anexo D – *Template User Story*

# Template – User Story

**CROSSING-PDS-2018-005**

**Versão 1.0**

**Desenvolvido por Filipe Rainho**

**Crossing Answers, Lda**

**2018**

# Histórico de revisões

| Nome | Data | Razão de Alterações | Versão |
|---|---|---|---|
| Filipe Rainho | 01/10/2018 | Versão inicial | 1.0 |
| | | | |
| | | | |
| | | | |
| | | | |

# Definição de User Stories

Uma User Story é uma tradução das necessidades do utilizador, incluindo os requisitos, e são apresentadas num formato que descreve o resultado da perspetiva do utilizador (utilizador pode ser interpretado como cliente final, cliente organizacional, etc.). Geralmente são a representação utilizada para os requisitos nos projetos desenvolvidos com base em abordagens ágeis.

As User Stories tipicamente têm origem numa Epic. No entanto podem ter origem numa outra User Story de maior dimensão. O principal fator de dimensão nas User Stories é deve ser possível terminar cada uma delas num único Sprint. Isto porque as User Stories são a base de estimativa para um Sprint (que são estimados em Story Points), permitindo assim saber a velocidade de uma determinada equipa num determinado projeto.

As User Stories são "propriedade" do Product Owner, que serve como cliente ou representante do cliente. O Producto Owner deve ajudar a manter as User Stories claras e concisas minimizando a necessidade de esclarecimentos à equipa e melhorando a sua eficiência durante os Sprints e na realização das estimativas. De forma a garantir a clareza nas User Stories devem ser definidos critérios de aceitação e devem existir canais de comunicação abertos com os *stakeholders*.

Na Tabela 1 é apresentada a listagem de atributos que uma User Story deve conter e também um exemplo prático para cada um dos atributos.

| # | Título | Descrição | Exemplo |
|---|---|---|---|
| 1 | ID | Identificador único para a User Story. É inserido automaticamente pelo sistema. | *Código único*. |
| 2 | Título | Cada User Story deve ter um título que, em resumo, explica a User Story. O título não deve ter mais do que uma frase. | Autenticação no sistema. |
| 3 | Descrição | A descrição contém informações mais detalhadas sobre a User Story. Além disso deve ser descrito o porquê e o benefício da sua implementação no sistema na ótica do negócio. | O sistema deve ser protegido por autenticação, não sendo possível aceder ao conteúdo sem uma sessão válida. A autenticação deve ser feita por email e palavra-passe, sendo ambos obrigatórios. No caso de serem fornecidas um par de credenciais válidas deve ser devolvida uma sessão no formato JWT com uma validade de 3 horas. O utilizador deve ser |

| | | | |
|---|---|---|---|
| | | | informado no caso de as credenciais fornecidas serem inválidas. |
| 4 | Critérios de aceitação | Cada User Story deve conter um conjunto de critérios de aceitação que permitem à equipa perceber todos os critérios que devem ser cumpridos na implementação da User Story. | • A autenticação deve ser feita com email e palavra-passe;<br>• As credenciais são obrigatórias no pedido de autenticação;<br>• Deve ser fornecido feedback ao utilizador em caso de credenciais inválidas;<br>• Em caso de credenciais válidas deve ser devolvida uma sessão no formato JWT;<br>• A sessão deve ter validade de 3 horas;<br>• Após terminar a validade da sessão o utilizador deve ser redirecionado para a vista de autenticação;<br>• O JWT fornecido deve ser armazenado na *store* do *browser*. |
| 5 | Relação | Deve existir uma relação com a Epic de onde teve origem o processo de refinação. | Feito no sistema de gestão de projeto através dos anexos. |
| 6 | Lista de tarefas | Quando a equipa procede ao desenvolvimento da User Story deve criar uma lista de tarefas com todos os passos necessários para o desenvolvimento da User Story. Acaba por ser um pequeno relatório das atividades realizadas. | • Desenvolvimento do *layout* do painel de autenticação;<br>• Validação local dos campos;<br>• Realização do pedido de autenticação ao servidor;<br>• Fornecer feedback ao utilizador consoante a resposta obtida;<br>• Armazenamento da sessão na *store* do |

| | | | |
|---|---|---|---|
| | | | *browser* em caso de sucesso. |
| 7 | Story Points | Número de Story Points atribuídos pela equipa no Backlog Grooming. Os Story Points utilizados devem respeitar a sequência de Fibonacci. | Número da sequência de Fibonacci (1, 2, 3, 5, 8, 13, etc.). |
| 8 | Plataforma | É representada por uma *tag* que classifica o tipo de plataforma em que o requisito irá ser desenvolvido. | DevOps, Back-end, Front-end, Mobile, Full Stack, etc. Todas as *tags* já existem no sistema de gestão de projeto. |
| 9 | Tempo gasto | Após a implementação deve ser feito o registo do tempo gasto na implementação da User Story em horas. | Por exemplo: 06:15 horas |
| 10 | Sprint e estado | Quando uma User Story é alocada a um Sprint deve ser registado qual o Sprint a que foi alocada tal como o estado em que se encontra. | Sprint X – Autenticação e Autorização.<br>Os estados possíveis são: Sprint Backlog; In Progress; Ready for Test; Done. |
| 11 | Criado por | O elemento que procedeu à definição e criação da User Story no Product Backlog. É inserido automaticamente pelo sistema. | *Nome*. |
| 12 | Data de criação | Data de criação. É inserido automaticamente pelo sistema. | |
| 13 | Data de atualização | Data da última atualização. É inserido automaticamente pelo sistema. | |

*Tabela 1 - Template de uma User Story*

De forma a trazer clareza ao processo de criação de User Stories podem ser usadas algumas técnicas para ajudar a trazer robustez ao Product Backlog, são elas:

- Considerar todas as fontes de informação pertinentes ao projeto e não apenas a informação disponibilizada pelo cliente. A perda de requisitos pode ter um impacto negativo no futuro.
- Definir que as User Stories têm uma prioridade que atende às necessidades do negócio e do cliente. Desta forma o *input* fornecido para o Backlog Grooming e para o Sprint Planning deve ser o mais assertivo possível.
- Garantir que as User Stories satisfazem os requisitos do utilizador no ambiente pretendido. Deve ser feito no início e muitas vezes no durante desenvolvimento para confirmar a viabilidade do sistema.
- Aplicar ciclos de feedback curtos de forma a garantir aprendizagem e que as User Stories presentes no Product Backlog estejam o mais robustas possível.

# Anexo E – Scrum

# Scrum

**CROSSING-PDS-2018-002**

**Versão 1.0**

**Desenvolvido por Filipe Rainho**

**Crossing Answers, Lda**

**2018**

# Histórico de revisões

| Nome | Data | Razão de Alterações | Versão |
|---|---|---|---|
| Filipe Rainho | 01/10/2018 | Versão inicial | 1.0 |
| | | | |
| | | | |
| | | | |
| | | | |

# Índice

# Definições e Acrônimos

A seguinte tabela apresenta os acrónimos usados ao longo deste documento.

| Acrónimo | Descrição |
|---|---|
| PO | Product Owner |
| SM | Scrum Master |
| ST | Scrum Team |
| | |
| | |

# Documentos Auxiliares

A seguinte tabela apresenta a lista de documentos auxiliares e documentos de referência do processo. Os documentos auxiliares e de referência são usados principalmente para fornecer mais leitura e informações complementares e/ou detalhadas.

| Referência | Documento |
|---|---|
| CROSSING-PDS-2018-004 | Template – Epic |
| CROSSING-PDS-2018-005 | Template – User Story |
| | |
| | |
| | |
| | |

# 1 Introdução

Este documento tem como principal objetivo detalhar a abordagem Scrum de forma a fornecer conhecimento e orientações para os colaboradores e *stakeholders* que tenham interesse em adquirir ou consolidar conhecimentos. É pretendido ajudar a melhorar a capacidade do público alvo de forma a produzir melhores resultados na utilização do Scrum.

O Scrum foi desenvolvido para gerir o processo de desenvolvimento de sistemas. É uma abordagem empírica que aplica as ideias da teoria de controlo de processos industriais ao desenvolvimento de sistemas, resultando numa abordagem que reintroduz as ideias de flexibilidade, adaptabilidade e produtividade. Não define nenhuma técnica específica de desenvolvimento de software. O Scrum concentra-se em como os membros da equipa devem trabalhar para desenvolver o sistema de forma flexível num ambiente em constante mudança.

A ideia principal do Scrum é que o desenvolvimento de sistemas envolve várias variáveis (por exemplo, requisitos, prazo, recursos e tecnologia) que provavelmente mudarão durante o processo. Isso torna o processo de desenvolvimento imprevisível e complexo, exigindo flexibilidade para que possa existir resposta à mudança. Como resultado, é produzido um sistema que é útil quando entregue ao cliente ou ao utilizador.

O Scrum ajuda a melhorar as práticas de engenharia existentes numa organização, uma vez que envolve atividades de gestão frequentes com o objetivo de identificar consistentemente quaisquer deficiências ou impedimentos no processo de desenvolvimento, bem como nas práticas utilizadas.

## 1.1 Estrutura do Documento

O documento está divido em sete seções, em seguida será feito um breve resumo do que cada uma delas irá abordar.

A primeira secção é a **Introdução**, e fornece informação sobre o documento, incluindo objetivos, propósito e também o público alvo do documento.

A segunda secção descreve as **Características e Valores** do Scrum enquanto uma *framework* de gestão de projetos ágil.

A terceira secção descreve o **Processo** do Scrum, sendo detalhado a que diz respeito cada uma das fases do processo.

A quarta secção descreve o **Ciclo do Scrum** e como devem ser colocadas em prática as suas cerimónias e utilizados os seus artefactos.

A quinta secção descreve os principais **Papeis e Responsabilidades** do Scrum, fazendo referência à sua obrigatoriedade e importância.

A sexta secção descreve as **Principais Cerimónias** do Scrum. É feita referência a como devem ser aplicadas e qual a intervenção de cada papel na cerimónia.

Por fim a sétima secção descreve os **Principais Artefactos** do Scrum. Tal como na secção anterior é explicado como devem ser utilizados e quais as responsabilidades de cada papel na sua utilização.

# 2 Características e Valores

O Scrum é uma *framework* de gestão de projetos ágil com o foco na entrega de maior valor de negócio no menor tempo possível. Permite a rápida e contínua inspeção e validação do software, isto porque as entre cada 1-4 semanas é possível um novo incremento no sistema em desenvolvimento. As prioridades do desenvolvimento são determinadas pelas necessidades do negócio e podem ser ajustadas a qualquer altura, mas apenas são consideradas na iteração a iniciar em seguida.

O Scrum concentra-se em como as equipas devem trabalhar para desenvolver o sistema de forma flexível num ambiente de constante mudança, isto introduzindo ideias de flexibilidade, adaptabilidade e produtividade.

A principais características do Scrum são as seguintes:

- Tem como base as técnicas e práticas do manifesto Ágil;
- Equipas autogeridas;
- O produto evolui numa serie de Sprints (iterações);
- Os requisitos são registados e documentados num Product Backlog;
- Não existem práticas de engenharia prescritas para a sua utilização.

Como referido anteriormente o Scrum tem como base as técnicas e práticas do manifesto Ágil, adotando os seu valores e princípios, sendo os valores presentes no manifesto Ágil os seguintes:

- **Indivíduos e iterações** mais do que processos e ferramentas.
- **Software funcional** mais do que documentação abrangente.
- **Colaboração com o cliente** mais do que negociação contratual.
- **Responder à mudança** mais do que seguir um plano.

Além dos valores adotados do manifesto Ágil, o Scrum também propõe a adoção dos seguintes valores por parte das equipas e das organizações:

- **Compromisso**
  - A Scrum Team (ST) compromete-se com um objetivo e define a melhor forma de o alcançar;
  - O Scrum Master (SM) compromete-se em manter o Sprint tal como for planeado;
  - O Product Owner (PO) compromete-se em manter o Product Backlog atualizado e priorizado.
- **Foco** – concentração no objetivo definido para o Sprint a decorrer.
- **Respeito**
- **Coragem** – a ST assume a responsabilidade de gestão do desenvolvimento.

# 3 O Processo

O Scrum inclui três fase no seu processo, todas elas devem ser realizadas e a sua ordem respeitada, essas três fases são:

1. Pre-game, onde é realizado o planeamento e definição da arquitetura de alto nível;
2. Development, que representa a parte ágil da abordagem Scrum e onde é desenvolvido o sistema;
3. Post-game, que diz respeito à conclusão da *release* e à entrega.

## 3.1 Pre-game

O planeamento inclui a definição do sistema que se pretende desenvolver. Deve ser criado um Product Backlog com todos os requisitos atualmente conhecidos. Os requisitos podem ter origem do cliente, do suporte ao cliente, de sessões de *brainstorming* da ST com os *stakeholders*, ou de qualquer outro *stakeholder* do projeto.

Os requisitos devem ser priorizados e o esforço estimado sempre que necessário. O Product Backlog deve constantemente atualizado como novos requisitos e alterações aos existentes, bem como com estimativas mais precisas e atualização da prioridade dos requisitos consoante as necessidades atuais do negócio.

O planeamento também inclui a definição da equipa para o projeto, as ferramentas e recursos necessários, a avaliação dos riscos, o cálculo da data de entrega, as necessidades de formação e a gestão da verificação e validação do sistema desenvolvido. A cada iteração o Product Backlog é revisto pela ST de forma a obter o seu compromisso para o Sprint seguinte.

A arquitetura de alto nível deve ser planeada com base no Product Backlog existente até ao momento. Num caso de melhoria de um sistema existente deve ser feita uma análise e revisão ao sistema de forma a identificar as alterações necessárias para implementar o Product Backlog definido. As decisões devem ser tomadas em equipa, sendo necessário chegar a um consenso entre todos os envolventes.

## 3.2 Development

Esta fase é tratada como uma "*black box*" onde as diferentes variáveis (como prazo, qualidade, requisitos, recursos, tecnologias, ferramentas de desenvolvimento e até métodos de desenvolvimento) identificadas, e que podem mudar durante o processo, são observadas e controladas através de várias práticas do Scrum durante os Sprints. Essas questões não são tidas em conta apenas no início do projeto, mas sim de forma regular para ser possível existir adaptação de forma flexível às mudanças que vão ocorrendo.

Na fase de desenvolvimento, o sistema é desenvolvido em Sprints. Os Sprints não são nada mais do que ciclos iterativos em que as funcionalidades são desenvolvidas ou aprimoradas para produzir novos incrementos no produto final. Cada Sprint incluí as tradicionais fases de

definição e análise de requisitos, arquitetura e design, codificação, testes e entrega. A arquitetura e design do sistema evoluem ao longo dos Sprints, sendo que cada um pode ter a duração de 1 a 4 semanas.

## 3.3 Post-game

Chega-se a esta fase quando o Product Backlog está implementado na totalidade e todos os problemas conhecidos resolvidos. Por norma nesta fase é planeado e executado o *deployment* do sistema. Pode incluir tarefas de integração, testes ao sistema e correção dos respetivos problemas encontrados e criação e disponibilização de documentação necessária.

# 4 O Ciclo do Scrum

Na Figura 1 é representado o ciclo do Scrum. O conjunto de atividades apresentado decorre durante a fase *Development*, sendo que deve ser respeitado ao máximo de forma a fornecer a todos os *stakeholders* do projeto o máximo de confiança possível durante o desenvolvimento do produto.

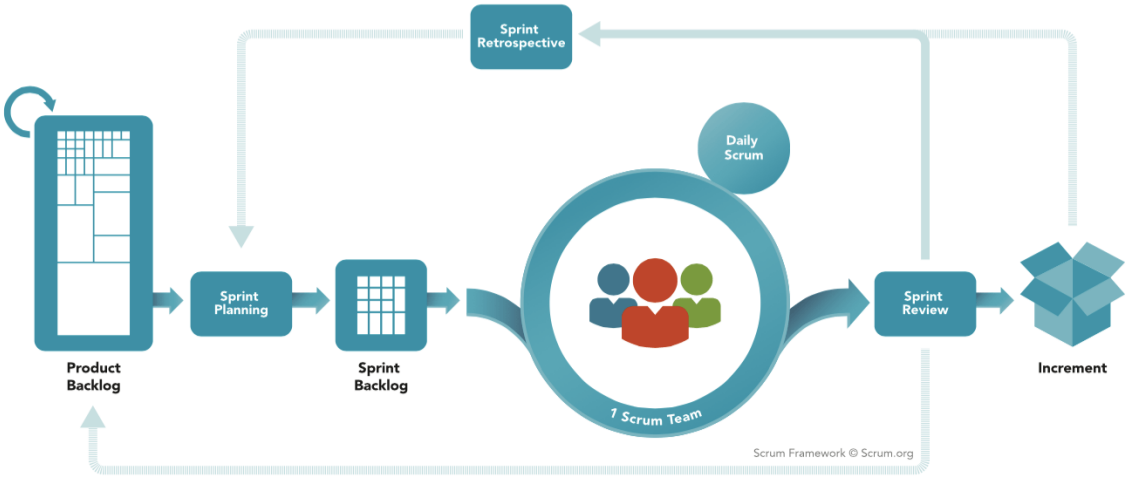



*Figura 1 - Ciclo do Scrum*

Como pode ser consulado na Figura 1 o ciclo inicia no Product Backlog, é no Product Backlog que se encontram definidos todos os requisitos conhecidos até ao momento. Os requisitos existentes vão sendo refinados, priorizados e geridos ao longo do tempo no Backlog Grooming. Apesar disso antes de cada Sprint é sempre realizado o Sprint Planning, onde também pode ser feita uma sessão de Backlog Grooming caso exista necessidade, mas o principal objetivo é a definição do Sprint Backlog. Todos os requisitos alocados ao Sprint Backlog devem ser claros o suficiente para a equipa e o somatório das suas estimativas não deve diferir muito da velocidade atual da equipa, isto porque é neste momento que a S cria o compromisso de entregar aquele conjunto de funcionalidade até ao final do Sprint.

Posto isto é iniciado o Sprint com a duração previamente definida no início do projeto e são executadas as tarefas de design e arquitetura, codificação e testes. Diariamente a equipa faz uma reunião chamada Daily Scrum Meeting, onde devem participar o PO, SM e a ST.

No final de cada Sprint deve ser feita a Sprint Review, onde são apresentadas as novas funcionalidades desenvolvidas e discutidos os passos seguintes. Além disso também deve ser realizada a Sprint Retrospective onde a equipa faz uma reflexão de como procedeu durante o Sprint e conclui com aspetos positivos e aspetos de melhoria. Por fim deve ser decido se o trabalho produzido no Sprint será integrado no sistema já existente (caso já exista em produção) ou se necessita de mais desenvolvimentos para ser integrado.

Este ciclo é repetido continuamente até o Product Backlog estar totalmente implementado ou até o tempo previsto para a *release* em questão chegar ao fim.

# 5 Papeis e Responsabilidades

O Scrum prescreve a definição de um conjunto de papeis e responsabilidades de caracter obrigatório aquando da sua utilização. Todos os intervenientes no desenvolvimento do sistema devem ter um dos seguintes cargos e responsabilidades:

- **PO:** o responsável por assegurar que a equipa de desenvolvimento oferece o maior valor possível ao cliente ou ao utilizador final no menor espaço de tempo. Desta forma, o PO tem responsabilidade de alimentar e gerir o Product Backlog, garantindo também a sua priorização e também de aceitar/rejeitar o resultado das funcionalidades desenvolvidas pela equipa;
- **SM:** encarregado por se certificar que os objetivos propostos para cada Sprint são realmente cumpridos conforme agendado, além disso garante a comunicação entre os diferentes papeis existentes sendo um "escudo" para a ST. Tem a responsabilidade de promulgar os valores e práticas do Scrum, garantindo a plena funcionalidade e produtividade da equipa;
- **ST:** composta pelos elementos (tipicamente 5-9 elementos) que trabalham em conjunto para desenvolver e entregar um incremento do produto ou sistema em cada Sprint. A ST deve ter a capacidade de autogestão e ser *cross-functional*.

Além dos cargos obrigatórios, os elementos da ST podem ter uma especificação ao nível das responsabilidades que vão ter no projeto com base num conjunto de cargos não menos importantes, mas de carácter opcional neste PDS:

- **Subject Matter Experts:** são elementos da equipa especialistas em categorias de competências específicas. Nesta categoria incluem-se elementos especialistas em linguagens de programação específicas, especialistas em UX, especialistas em controlo e gestão de bases de dados, entre outros. No caso de ser identificada esta necessidade, a sua especialidade deverá ser devidamente anotada;
- **Software Quality Engineer:** elementos responsáveis por assegurar a qualidade do código produzido durante cada Sprint, criar casos de teste e desenvolver testes automatizados e manuais para testar cada um dos requisitos por completo;
- **Documentation Specialist:** este especialista é responsável por criar os manuais de utilizador, manuais de administrador, e outros documentos de treino essenciais à distribuição e comercialização do produto ou sistema desenvolvido.

Por fim, o papel desempenhado pelo tanto pelo cliente como pela gerência devem ser considerados. As suas principais responsabilidades são:

- **Cliente**: participa em todas as tarefas relacionadas com a gestão e manutenção do Product Backlog e no fornecimento de feedback no final de cada Sprint.
- **Gerência**: é responsável pela tomada de decisões, participa na definição de metas e requisitos e é responsável pela escolha do PO.

# 6 Principais Cerimónias

Como já referido o Scrum não prescreve quaisquer práticas de engenharia ou de desenvolvimento de software específicos. Contrariamente, requer a adoção de cerimónias e a utilização de alguns artefactos de forma a evitar ao máximo o caos causado pela imprevisibilidade e complexidade dos sistemas desenvolvidos.

Nesta secção são apresentadas as principais cerimónias a considerar quando se aplica o Scrum como base do processo de desenvolvimento de software de uma organização. As principais cerimónias são:

- Backlog Grooming
- Sprint Planning
- Daily Scrum Meeting
- Sprint Review
- Sprint Retrospective

## 6.1 Backlog Grooming

O Backlog Grooming é uma técnica ágil amplamente utilizada em processos com base no Scrum de forma a produzir um Product Backlog priorizado antes e durante os Sprints.

As principais valências do do Backlog Grooming são a produção de um Product Backlog priorizado e atualizado, isto porque durante esta cerimónia é incluída a negociação entre o PO e a ST sobre as Epics e User Stories que serão adicionadas, modificadas, removidas e priorizadas no Product Backlog. Apesar disso todos os stakeholders relevantes devem contribuir para essa decisão colaborativa.

O PO é o principal proprietário do Product Backlog. A qualquer momento novas Epics e User Stories podem surgir como resultado de contribuições *stakeholders*. É responsabilidade do PO promover o Backlog Grooming de forma a registá-las no Product Backlog juntamente com os respetivos critérios de aceitação. Isto porque o uso de critérios estabelecidos para criar ou alterar User Stories irá fortalecer os Sprint Backlogs e otimizar as Sprint Riviews.

## 6.2 Sprint Planning

O Sprint Planning ocorre no início de cada Sprint, e é uma negociação entre a ST e o PO no que diz respeito ao valor será entregue no próximo Sprint. Durante o Sprint Planning, é desenvolvido o Sprint Backlog e são identificadas as tarefas para suportar as User Stories planeadas. Os membros da equipa avaliam quanto trabalho podem realizar durante o Sprint e são designados para as várias tarefas a serem concluídas. O Sprint Planning deve ocorrer sempre no início de cada Sprint, dependendo da duração entre 1-4 semanas, e deve identificar apenas o valor a ser entregue durante o Sprint em questão.

Ainda no Sprint Planning devem ser feitas considerações de planeamento e arquitetura de forma garantir o entendimento e coerência entre as implementações dos elementos da ST.

A cerimónia Sprint Planning é muito importante porque permite à ST ter conhecimento dos seguintes fatores:

- Deve saber-se no que se irá trabalhar;
- Deve perceber-se o suficiente para o fazer.

## 6.3 Daily Scrum Meeting

A Daily Scrum Meeting é uma reunião diária, com uma duração de no máximo 15 minutos e onde todos os elementos devem estar de pé. Esta cerimónia é usada de forma a identificar problemas e riscos mais cedo do que num projeto tradicional ("*fail fast*") e aumentar a colaboração entre os membros da ST. Permite também evitar reuniões adicionais e desnecessárias, mas não tem como objetivo a resolução de problemas. Na reunião todos os participantes autorizados a falar devem responder a três questões:

- O que fiz ontem?
- O que vou fazer hoje?
- Com que problemas me deparei e estão no meu caminho?

Durante a reunião, os elementos colaboram de forma a reagir melhor aos problemas e riscos identificados. Toda a gente pode participar na reunião, mas apenas a ST, SM e PO podem falar.

## 6.4 Sprint Review

O Sprint Review é uma técnica colaborativa e incremental para garantir que todos os *stakeholders* estejam cientes do valor que está a ser entregue no final de cada Sprint para ST. Durante o Sprint Review, o trabalho que foi concluído e o trabalho que foi planeado sofre uma revisão. A apresentação do trabalho concluído aos *stakeholders* é chamado de Sprint Demo, podendo ser apresentadas novas funcionalidades desenvolvidas e arquitetura subjacente.

O Sprint Review ocorre no final de cada Sprint e imediatamente antes da Sprint Retrospective. Todos podem assistir, mas é importante contar com todos os *stakeholders* relevantes. Desta forma, o Sprint Review pode ser considerado uma forma de comunicação, validação e reconhecimento de que a ST atingiu o objetivo pretendido.

## 6.5 Sprint Retrospective

A Sprint Retrospective é uma reunião que deve ser realizada no final de cada Sprint, onde todos podem participar e deve ter um máximo de 30 minutos. Tem como principal objetivo perceber o que está e o que não está a funcionar.

Todos os presentes, em conjunto, devem responder a duas questões:

- O que fizemos bem?
- O que devíamos ter feito melhor?

Os registos criados ao longo do tempo irão permitir criar uma base de informação muito útil para a otimização de processos nas organizações.

# 7 Principais Artefactos

Nesta secção são apresentados os principais artefactos a considerar quando se aplica o Scrum como base do processo de desenvolvimento de software de uma organização. Os principais artefactos são:

- Epics
- User Stories
- Product Backlog
- Sprint Backlog
- Sprint Burndown Chart

## 7.1 Epics

Uma Epic é uma grande User Story que exigirá uma análise mais detalhada e que será necessário refinar em User Stories mais pequenas antes de serem alocada a um Sprint. Esta escolha é feita pelas equipas de forma a facilitar a gestão dos requisitos e ser possível completar as User Stories num único Sprint.

As Epics são um *work product* criado como resultado do processo de eliciatação e desenvolvimento de requisitos. Posteriormente são refinadas em User Stories durante o Backlog Grooming, ou durante o planeamento de uma *release*, ou de forma contínua pelo Product Owner ao longo do desenvolvimento do projeto. Este processo é geralmente iterativo.

O PO pode apresentar um requisito à equipa achando que se trata de uma User Story, apesar disso a equipa pode considerar que é uma Epic e fazer uma maior divisão. Não existe uma medida para diferenciar uma Epic de uma User Story, embora seja comum considerar que uma User Story pode abranger vários Sprints, mas uma Epic não. A alocação aos Sprints não é a única razão para refinar as Epics, um outro motivo é a clareza que o requisito necessita para ser desenvolvido.

Para consultar mais informação relativa à definição de Epcis deve ser consultado o anexo [CROSSING-PDS-2018-004].

## 7.2 User Stories

Uma User Story é uma tradução das necessidades do utilizador, incluindo os requisitos, e são apresentadas num formato que descreve o resultado da perspetiva do utilizador (utilizador pode ser interpretado como cliente final, cliente organizacional, etc.). Geralmente são a representação utilizada para os requisitos nos projetos desenvolvidos com base em abordagens ágeis.

As User Stories tipicamente têm origem numa Epic. No entanto podem ter origem numa outra User Story de maior dimensão. O principal fator de dimensão nas User Stories é deve ser possível terminar cada uma delas num único Sprint. Isto porque as User Stories são a base de

estimativa para um Sprint (que são estimados em Story Points), permitindo assim saber a velocidade de uma determinada equipa num determinado projeto.

As User Stories são "propriedade" do PO, que serve como cliente ou representante do cliente. O PO deve ajudar a manter as User Stories claras e concisas minimizando a necessidade de esclarecimentos à equipa e melhorando a sua eficiência durante os Sprints e na realização das estimativas. De forma a garantir a clareza nas User Stories devem ser definidos critérios de aceitação e devem existir canais de comunicação abertos com os *stakeholders*.

Para consultar mais informação relativa à definição de User Stories deve ser consultado o anexo [CROSSING-PDS-2018-005].

## 7.3 Product Backlog

O Product Backlog é uma lista priorizada de tudo o que pode ser incluído no projeto a desenvolver. Inclui funcionalidades (tipicamente em formato de Epics e User Stories), alterações na documentação e quaisquer outras tarefas exigidas pelo PO.

A gestão e priorização do Product Backlog pertencem ao PO. O PO pode usar as informações dos *stakeholders*, representantes comerciais e da ST para ajudar a definir a prioridade, mas é o PO o responsável. O Product Backlog é mantido durante os Sprints através do Backlog Grooming.

O Product Backlog é a base de um projeto desenvolvido com uma abordagem ágil, independentemente de qual seja. O seu conteúdo deve ter valor para o cliente/utilizador.

## 7.4 Sprint Backlog

O Sprint Backlog é um conjunto de User Stories selecionadas pela ST para serem desenvolvidas num Sprint. Deve representar a previsão da ST sobre qual será o próximo incremento no sistema devendo ser definido o objetivo para o incremento (Sprint Goal).

O Sprint Backlog é propriedade da ST e é contruído através de um processo de negociação entre a ST e o PO, após a negociação é criado um compromisso para aquele Sprint. O Sprint Backlog tonar-se a imagem do trabalho que a ST pretende produzir durante o Sprint.

Existem algumas regras que devem ser respeitadas pelos intervenientes no projeto relativamente ao Sprint Backlog, são elas:

- Não deve ser alterado até ao final do respetivo Sprint;
- É propriedade da ST, sendo os mesmos responsáveis pela sua gestão;
- O trabalho restante deve ser atualizado diariamente (por exemplo através da utilização de um Burndown Chart);
- Todas as User Stories selecionadas para o Sprint Backlog devem ser claras o suficiente para a ST as conseguir implementar.

## 7.5 Sprint Burndown Chart

Um Sprint Burn-Down Chart é um gráfico que representa visualmente o esforço restante para terminar o Sprint. Ajuda a equipa a entender se será possível fornecer as funcionalidades desejadas até o final do Sprint, conforme o planeado.

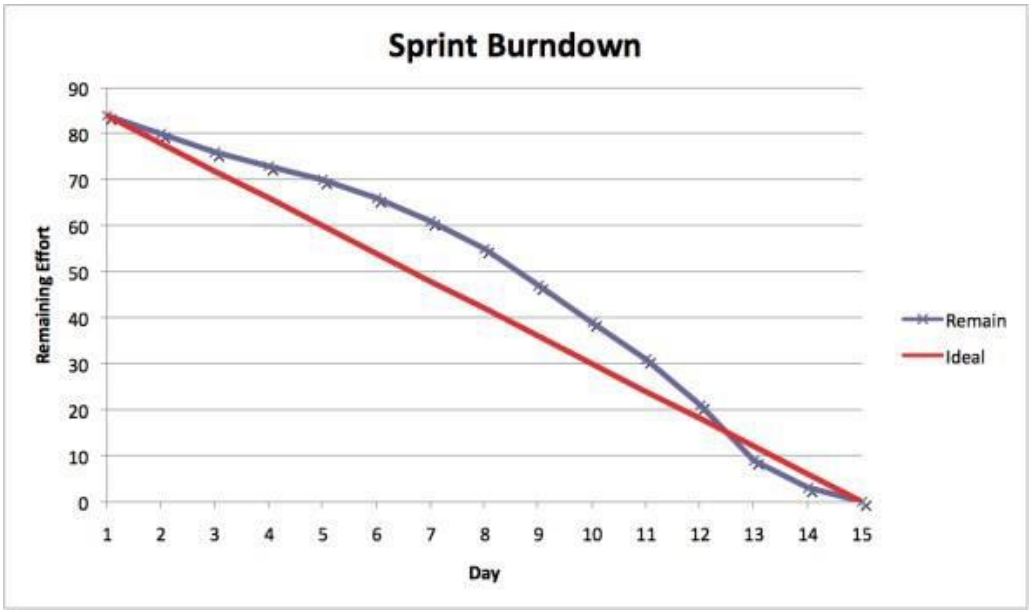



*Figura 2 - Sprint Burndown Chart*

Na Figura 2 é apresentado o progresso de um ST em relação ao final de um Sprint, não em termos de tempo gasto, mas de quantidade de trabalho realizado. É possível a qualquer altura perceber o trabalho que foi feito e o que falta ser feito.

O mesmo tipo de gráfico foi ser aplicado à *release* atual, sendo representada a trajetória por cada Sprint e com base no número de Story Points que a equipa é capaz de "queimar" por Sprint, será possível entender se irão ou não entregar as funcionalidades desejadas no período de tempo planeado.

# Anexo F – Kanban

# Kanban

**CROSSING-PDS-2018-003**

**Versão 1.0**

**Desenvolvido por Filipe Rainho**

**Crossing Answers, Lda**

**2018**

# Histórico de revisões

| Nome | Data | Razão de Alterações | Versão |
|---|---|---|---|
| Filipe Rainho | 01/10/2018 | Versão inicial | 1.0 |
| | | | |
| | | | |
| | | | |
| | | | |

# Índice

# Definições e Acrônimos

A seguinte tabela apresenta os acrónimos usados ao longo deste documento.

| Acrónimo | Descrição |
|---|---|
| JIT | *just-in-time* |
| WIP | *work in progress* |
| | |
| | |
| | |

# Documentos Auxiliares

A seguinte tabela apresenta a lista de documentos auxiliares e documentos de referência do processo. Os documentos auxiliares e de referência são usados principalmente para fornecer mais leitura e informações complementares e/ou detalhadas.

| Referência | Documento |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |

# 1 Introdução

Este documento tem como principal objetivo detalhar a abordagem Kanban de forma a fornecer conhecimento e orientações para os colaboradores e *stakeholders* que tenham interesse em adquirir ou consolidar conhecimentos. É pretendido ajudar a melhorar a capacidade do público alvo de forma a produzir melhores resultados na utilização do Kanban.

O kanban é a palavra japonesa para "sinal visual" ou "cartão". Nasceu em 1940 na Toyota sendo a fonte de inspiração a lógica de funcionamento dos supermercados, onde as reposições são feitas tendo em conta o inventário e não o que os fornecedores fornecem. Só quando o produto está prestes a ficar esgotado é que é pedido mais, desta forma os níveis de stock correspondem aos padrões de consumo, garantindo eficiência na gestão de stocks e diminuindo o armazenamento do excesso de stock.

O processo de entrega *just-in-time* (JIT) levou os engenheiros da Toyota a repensar os seus métodos e a abrir caminho para uma nova abordagem, o Kanban. Os trabalhadores passaram a usar o Kanban para gerir as etapas do processo de fabricação. A natureza visual do sistema permitiu que as equipas comunicassem mais facilmente sobre o trabalho que era necessário fazer e quando seria necessário fazê-lo.

As equipas ágeis de desenvolvimento de software são capazes de alcançar os princípios JIT, combinando a quantidade de *work in progress* (WIP) com a capacidade da equipa. Desta forma são proporcionadas opções de planeamento mais flexíveis, entregas mais rápidas, foco mais claro e transparência em todo o ciclo de desenvolvimento.

De forma geral, a utilização do Kanban requer capacidade de comunicação em tempo real e total transparência em todo o processo. Os *work items* devem ser representados visualmente num Kanban Board, permitindo que os membros da equipa vejam o estado de cada parte do trabalho a qualquer momento.

## 1.1 Estrutura do Documento

O documento está divido em sete seções, em seguida será feito um breve resumo do que cada uma delas irá abordar.

A primeira secção é a **Introdução**, e fornece informação sobre o documento, incluindo objetivos, propósito e também o público alvo do documento.

A segunda secção descreve os **Princípios** base do Kanban explicando o motivo da sua existência.

A terceira secção enumera os **Benefícios e Desvantagens** na utilização do Kanban para o desenvolvimento de projetos.

A quarta secção descreve o **Processo e o Ciclo de Vida** do Kanban. Apesar de ser pouco prescritivo é necessário seguir alguns pontos chave.

Por fim a quinta secção apresenta uma comparação, **Kanban VS Scrum**.

# 2 Princípios

O kanban procura identificar oportunidades de melhoria criando uma cultura de melhoria contínua do processo na qual a responsabilidade é de todos. Os princípios básicos são:

- Visualização completa da cadeia de valor;
- Limitar a quantidade de WIP;
- Ajudar a melhorar o *workflow*;
- Promover o desenvolvimento incremental, evolutivo e adaptativo.

Dada a origem da palavra, não é surpreendente que o primeiro princípio do Kanban seja relativo à visualização. Contrariamente a outras abordagens, o Kanban não prescreve um fluxo de trabalho, requer apenas que seja documentado de uma maneira que possa ser facilmente visualizado e que seja possível ver todos os *work items* no contexto correto. Após o mapeamento visual do processo atual, as oportunidades de melhoria começam a ficar mais obvias.

O objetivo do Kanban é realizar cada trabalho com máximo de eficiência desde o início até ao fim com o mínimo de desperdício possível. Isso requer limitar a quantidade de trabalho no *workflow* para o máximo que pode ser gerido num determinado momento. Permite as equipas não criarem compromisso com muito trabalho em simultâneo.

Tendo os dois primeiros princípios em vigor, o trabalho flui mais livremente e de forma mais fácil, isto é, quando algo é concluído o próximo *work item* no Product Backlog é selecionado. Sendo assim a atenção passa a estar virada para possíveis interrupções do fluxo. Esses acontecimentos representam oportunidades para a melhoria do processo.

O Kanban não é algo que está "terminado". A abordagem requer monitorização e análises constantes de forma a promover otimizações. As condições, recursos e procura mudam com o tempo, por isso é sempre importante avaliar o *work flow* e procurar por possíveis problemas que consigam ser evitados.

# 3 Benefícios e Desvantagens

O Kanban é uma das metodologias de desenvolvimento de software mais populares e, atualmente, adotadas por equipas ágeis. Oferece vários benefícios para o planeamento de tarefas para equipas de todos os tamanhos. Os principais benefícios do Kanban são:

- Versatilidade;
- Melhoria contínua;
- Responsividade à procura;
- Aumento na produtividade;
- *Empowerment*;
- Qualidade.

Tal como qualquer metodologia ou abordagem a sua utilização também apresenta algumas desvantagens que serão apresentadas nesta secção.

## 3.1 Versatilidade

A ideia central por trás do Kanban é comunicar através de sinais visuais, cujos os benefícios abrangem indústrias e cargos. Sendo universalmente aplicável, o Kanban pode ser implementado por todas as equipas de uma organização (engenharia, marketing, administração, etc.). A sua adoção é mais simples do que grande parte de outras abordagens porque requer menos alterações de configuração na organização para começar a usar.

## 3.2 Melhoria contínua

Todos os elementos devem estar focados na melhoria contínua. O sistema visual do Kanban facilita a revisão de processos aplicando melhorias que agilizam o *workflow* e reduzem/anulam o *overworking* nas equipas. Permite reduzir o desperdício e remover atividades que não agregam valor para a organização.

## 3.3 Responsividade à procura

Na sua origem, o Kanban foi criado para adequar o stock à procura, acionando o processo apenas quando o stock estava muito em baixo (entrega JIT). Desta forma permite melhorar a capacidade de resposta à mudança das necessidades do negócio, sendo ideal para quando as prioridades mudam com muita frequência.

## 3.4 Aumento na produtividade

O Kanban incentiva as equipas a limitarem o WIP, criando uma *pipeline* de trabalho sustentável. Limitar o WIP incentiva as equipas a trabalharem juntas para fazer chegar o trabalho ao estado de entrega, eliminando o máximo de distrações causadas pelo *multitasking*.

Os ciclos de *feedback* rápido e a colaboração melhoram as hipóteses dos membros das equipas em termos de motivação, capacidade e desempenho. Os limites de WIP podem ser ajustados e os seus efeitos medidos.

## 3.5 Empowerment

Todos os elementos da equipa têm acesso à Kanban Board, partilhando a responsabilidade de fazer chegar o trabalho ao estado de entrega. O Kanban capacita as equipas para conseguirem tomar decisões mais ágeis que impulsionam a eficiência e produtividade na organização. As equipas gerem a execução das suas tarefas diárias, isto permite aos gestores ganharem tempo para se concentrarem no planeamento de longo prazo.

Um outro grande benefício é a resistência à mudança porque os colaboradores participam diariamente no processo de decisão.

## 3.6 Qualidade

O sistema do Kanban prevê a utilização de pequenos lotes de *work items* em várias fases da produção, desta forma os problemas podem ser identificados na sua origem. E ainda com a limitação do WIP é excluída a possibilidade de existirem problemas por identificar durante longos períodos de tempo.

Os ciclos de tempo mais curtos podem fornecer requisitos mais rapidamente e ajustar os mesmos às necessidades do negócio e o foco na melhoria contínua e na capacidade de resposta aos problemas significa que os projetos, até a sua conclusão, podem ter menos erros e correções.

## 3.7 Desvantagens

Como referido o Kanban apresenta também algumas desvantagens e limitações, em seguida são apresentadas algumas delas:

- Se um elemento da equipa criar um *bottleneck* no *worlflow*, para proceder à sua resolução é necessário que todos os elementos tenham conhecimento dos *work items* a decorrer e também que exista clareza na definição dos mesmos;
- Só funciona se os *work items* puderem ser separados muito claramente uns dos outros e divididos em etapas de trabalho individuais;
- Projetos com um prazo fixo são melhor geridos por outras *frameworks* como o Scrum, que colocam um maior foco na gestão do tempo.

# 4 Processo e Ciclo de Vida

O kanban pode ser utilizado em diversas áreas, como por exemplo, desenvolvimento de software, design, finanças, marketing, gestão operacional, entre outras. O uso adequado do Kanban pode trazer agilidade e organização. Agilidade em responder às mudanças que surgem ao longo da execução de tarefas e projetos, e organização, formando uma equipa que trabalha de forma mais eficiente.

O processo e ciclo de vida do Kanban tem como base os quatro principais passos, são eles:

1. Equipa
2. Identificação dos estados do trabalho
3. Priorização
4. Medição e melhoria contínua

## 4.1 Equipa

É importante que a equipa esteja preparada para assimilar os conceitos e princípios do Kanban. Pode haver resistência, uma vez que a transparência traz como consequência a visualização de quem é produtivo e também de quem não é. Desta forma é muito importante apresentar os conceitos, princípios, e um exemplo básico de uso do Kanban representando assim os benefícios que a equipa poderá obter com o seu uso.

É sempre bom lembrar que, independentemente da ferramenta, a equipa é a parte mais importante. Uma equipa com capacidades funciona bem utilizando qualquer *framework*, já uma equipa menos capaz geralmente falha na utilização até mesmo dos melhores métodos.

Tendo entendido o básico, é possível construir o Kanban Board que mais irá atender as necessidades atuais da equipa.

## 4.2 Identificação dos estados do trabalho

A identificação dos estados do trabalho é uma tarefa muito importante na preparação da Kanban Board isto porque vai ter influência direta no *workflow* que a equipa vai seguir. O fluxo mais comum é apresentado na Figura 1.

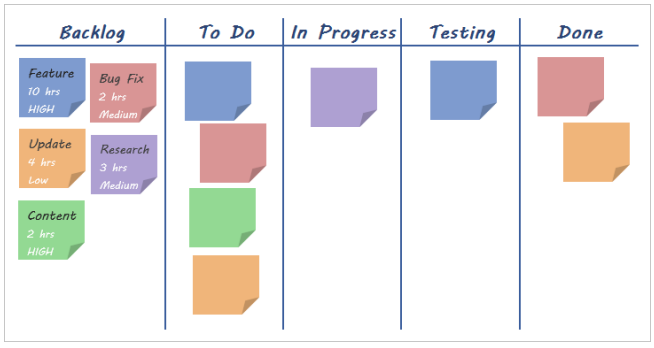



*Figura 1 - Kanban Board workflow*

Caso a equipa utilize a mesma Kanban Board para desenvolvimento e para suporte ao sistema que se encontra em produção as diferentes classes devem ser identificadas (por exemplo: User Story, *bug*, etc.), isto porque vai ajudar a equipa a categorizar o trabalho e a ter uma Kanban Board mais organizada.

O WIP pode ser limitado por cada estado da Kanban Board. Sendo permitidas alocações de novas funcionalidades a qualquer altura desde que não seja excedido o limite para o estado em questão. Na Figura 2 é apresentada uma representação de um cenário em que o limite é atingido foi criado um *bottleneck*.

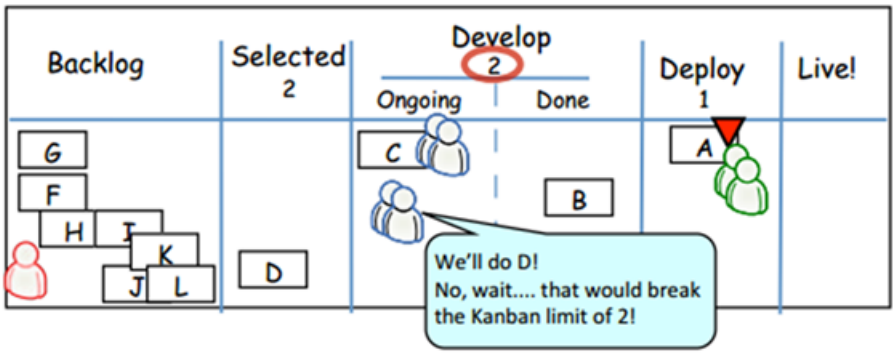



*Figura 2 - Limite de WIP*

No caso da Figura 2 foi criado um *bottleneck* em que não é possível iniciar o desenvolvimento de uma nova tarefa até ser libertado um dos *work items* pendentes (na figura o C e o B). Esta situação pode ter consequências de maior dimensão caso surja um relatório de um problema crítico na versão do sistema que está em produção e ao qual não é possível atender neste momento.

Para tal as Kanban Boards devem ter uma *priority lane* ou *fast lane* que percorre todos os estados da Kanban Board e tem um limite próprio para o WIP. É reservada para a resolução de problemas críticos e inesperados, com já referido. Na Figura 3 é feita a sua representação.

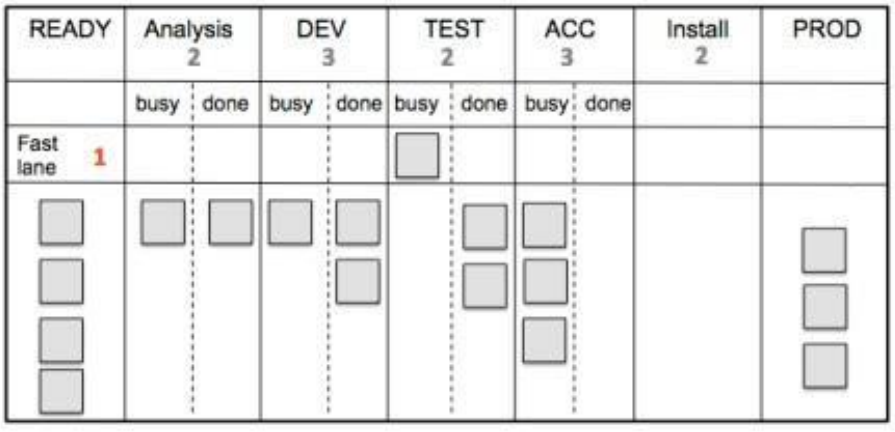



*Figura 3 - Kanban Board fast lane*

Existem algumas regras na utilização da *priority lane*, nomeadamente evitar a sua obstrução com *work items* que não são críticos, isto porque, por norma, é limitada a um *work item* em todo o seu fluxo.

## 4.3 Priorização

A priorização é especialmente importante, porque permite manter o foco da equipa no que realmente se deve fazer em primeiro lugar, e que por sua vez vai ter maior valor para o cliente.

Tal como a existência de um Product Backlog é opcional no Kanban, também não existe uma prescrição para a priorização. No entanto é recomendado que os *work items* mais prioritários sejam posicionados na parte superior da Kanban Board. Uma outra prática comum é a utilização de códigos de cor para definir a prioridade dos *work items*. Deve ser mantido um controlo constante do que é realmente prioritário, especialmente quando é feita a entrada de novos *work items* na pilha. Desta é possível melhorar a qualidade e reduzir os custos, eliminado/adiando o trabalho desnecessário.

A priorização ajuda a evitar os seguintes problemas:

1. Entrega de software inútil;
2. Cliente insatisfeito;
3. Metas não atingidas;
4. Demora em entregar *bugs* corrigidos e melhorias implementadas.

## 4.4 Medição e melhoria contínua

Estando o processo em vigor, torna-se a base para uma cultura de melhoria contínua. As equipas devem medir a sua eficiência acompanhando o *workflow*, a qualidade, a produtividade e os prazos de entrega.

Para atingir a melhoria contínua, adaptação é a palavra-chave. A Kanban Board definida inicialmente não vai permanecer igual para sempre, é necessário ser recetivo à mudança e ter uma mente aberta o suficiente para identificar oportunidades de melhoria. Ao longo do tempo será necessário adaptar o *workflow* com mais ou menos estados consoante o projeto ou então definir novas regras de priorização de forma a ser compatível com o negócio em questão.

É importante estabelecer um sistema de medição simples que permita avaliar o trabalho que precisa de ser feito e o que já foi feito (no caso do Scrum é utilizado o Burndown Chart).

Deve existir preocupação em simular risco durante a execução do trabalho, procurando encontrar *bottlenecks* antes que os mesmos realmente ocorram. Desta forma é possível determinar planos de ação preventivos, que diminuem consideravelmente os riscos.

Concluindo, as experiências e as análises podem provocar alterações para melhorar a eficácia das equipas.

# 5 Kanban VS Scrum

Nesta secção é feita a comparação entre o Kanban e o Scrum. São apresentadas as semelhanças e diferenças entre ambos, de forma a auxiliar os colaboradores a conseguirem utilizar ambos consoante o a escolha feita para o projeto.

As principais semelhanças entre o Kanban e o Scrum são as seguintes:

- Ambos são abordagens do Ágil;
- Ambos são baseados em *pull scheduling*;
- Ambos limitam o WIP;
- Ambos usam transparência para melhorar o processo;
- Ambos se concentram em fornecer software com antecedência e frequência;
- Ambos são baseados em equipas auto-organizadas;
- Em ambos o *release plan* é continuamente otimizado com base em dados empíricos (Velocidade e Lead Time, respetivamente).

Por fim as principais diferenças entre ambos são apresentadas na Tabela 1.

| **Scrum** | **Kanban** |
|---|---|
| Iterações *timeboxed* | Não prescreve nenhuma prática |
| Equipa compromete-se com uma quantidade de trabalho por iteração | Compromisso é opcional |
| Velocidade é a métrica para o planeamento | Lead Time é a métrica para o planeamento |
| Equipas *cross-functional* | Opcional |
| Requisitos são subdivididos para ser alocados a uma iteração | Não prescreve nenhuma prática |
| WIP é limitado por iteração | WIP é limitado por estado |
| Iterações são fechadas a novos *work items* | Podem ser adicionados novos *work items* a qualquer altura tendo em conta a capacidade |
| Sprint Backlog pertence à equipa | Kanban Board pode ser partilhada por diferentes equipas |
| Prescreve três roles: Product Owner; Scrum Master; Scrum Team; | Não prescreve nenhuma prática |
| Sprint Backlog é criado para cada iteração | Kanban Board é persistido |
| Prescreve um Product Backlog priorizado | A priorização é opcional, tal como o método escolhido para a mesma |

*Tabela 1 - Scrum VS Kanban*

# Anexo G – *Template Bug*

# Template - Bug

**CROSSING-PDS-2018-006**

**Versão 1.0**

**Desenvolvido por Filipe Rainho**

**Crossing Answers, Lda**

**2018**

# Histórico de revisões

| Nome | Data | Razão de Alterações | Versão |
|---|---|---|---|
| Filipe Rainho | 01/10/2018 | Versão inicial | 1.0 |
| | | | |
| | | | |
| | | | |
| | | | |

# Definição de Bugs

Um problema é uma propriedade questionável na execução de um sistema. Torna-se um *bug* se tiver um comportamento incorreto ou um pedido de melhoria se for algo que esteja ausente. Quando o utilizador informa sobre um problema é realizado o seguinte ciclo:

1. Reproduzir o problema;
2. Isolar as circunstâncias;
3. Localizar e corrigir o defeito;
4. Entregar a correção ao cliente.

Este documento é direcionado à gestão dos *bugs*. A organização do ciclo de vida é muito importante, para tal deve ser possível responder facilmente às seguintes questões:

- Quais os problemas em aberto atualmente?
- Quais os problemas mais graves?
- Ocorreram problemas semelhantes no passado?

Todos os *bugs* devem dar origem a um cartão na ferramenta de gestão de projetos. Nesse cartão devem ser incluídas todas as informações necessárias para corrigir o *bug*, para isso é imprescindível saber o que reportar. A informação a reportar é dividida em dois grandes grupos:

- Factos do problema (passos para reproduzir o problema encontrado, comportamento experienciado e o esperado, resumo com a informação essencial do problema, etc.);
- Factos do produto (versão, sistema operativo, ambiente de execução, etc.).

Atualmente os *bugs* são classificados segundo vários fatores, são eles a severidade, prioridade e tipo de plataforma. A prioridade dos *bugs* é representada através da ordenação dos mesmos no Bug Backlog, sendo a ordenação feita do mais prioritário para o menos prioritário. O tipo de plataforma é representado por uma *tag* (tal como as User Stories) e tem como objetivo ajudar a análise em projetos que tem desenvolvimentos em vários elementos da pilha de desenvolvimento (por exemplo: Back-end, Front-end, Mobile, etc.). Por fim, quanto à severidade existem quatro categorias para classificar os *bugs*, são elas:

- **Minor:** problemas simples que não ocorrem sempre e quando ocorrem não impedem o funcionamento do sistema ou questões relativas ao layout de uma vista (por exemplo: alinhamento de componentes, animações, etc.).
- **Normal:** problema que impede a utilização de uma funcionalidade do sistema, mas apenas num cenário específico (por exemplo: falta de validação ou validação errada de um campo, a pesquisa por um determinado campo não funcionar, etc.).
- **Important:** problema que impede a utilização de uma funcionalidade do sistema, mas não o sistema na totalidade (por exemplo: um formulário de inserção na inserção o registo, a funcionalidade de pesquisa numa lista não funcionar de todo, etc.).
- **Critical:** problema que impede a utilização de todo o sistema, ou seja, mesmo não existindo problemas noutras funcionalidades o problema em questão afeta todas

as outras (por exemplo: não ser possível carregar o sistema, ou o sistema não passar da autenticação, etc.) ou então a funcionalidade afetada é tão necessária que não faz sentido usar o sistema sem ela (por exemplo: um sistema de faturação não deixar emitir faturas, ou um chat não permitir enviar/receber mensagens).

Como referido anteriormente, todos os *bugs* são registados e armazenados no Bug Backlog. Quando a equipa pretende proceder à resolução de um ou vários *bugs* de movê-los para a *board* de resolução de bugs da Collection do projeto. Esta *board* tem o formato de uma Kanban Board e representa os diferentes estágios que um *bug* percorre quando é alocado para resolução. Os estados são: New; In Progress; Ready for Test; Done. Todos os *bugs* devem ser resolvidos com base neste fluxo de forma a permitir à gestão de projeto perceber o estado de cada um deles.

Na Tabela 1 é apresentada a listagem de atributos que um *bug* deve conter e também um exemplo prático para cada um dos atributos.

| # | Título | Descrição | Exemplo |
|---|---|---|---|
| 1 | ID | Identificador único para a User Story. É inserido automaticamente pelo sistema. | *Código único*. |
| 2 | Título | Cada *bug* deve ter um título que, em resumo, explica o *bug*. O título não deve ter mais do que uma frase. | A vista de listagem de projetos não faz *scroll* em dispositivos iPhone. |
| 3 | Descrição | A descrição contém informações mais detalhadas sobre o *bug*. Deve ser considerado apresentar os passos para reproduzir o problema e o que foi experienciado e o esperado. | A listagem de projetos deveria fazer *scroll* vertical devido ao elevado número de projetos, no entanto o *scroll*, apesar de estar presente, não tem qualquer reação aos gestos do utilizador. Esta situação acontece em qualquer *browser,* mas apenas em dispositivos da marca iPhone. |
| 4 | Versão do sistema | Verão do sistema em que o *bug* é acionado. Se possível referenciar a data relativa à versão. | V2.1 de XX-XX-XXXX. |
| 5 | Plataforma | É representada por uma *tag* que classifica o tipo de plataforma em que o requisito irá ser desenvolvido. | Back-end, Front-end, Mobile, Full Stack, etc.<br>Todas as *tags* já existem no sistema de gestão de projeto. |
| 6 | Severidade | É representada por uma *tag* e pretende mostrar qual a severidade do bug na perspetiva no negócio. | As severidades possíveis são: Minor; Normal; Important; Critical. |

| | | | |
|---|---|---|---|
| 7 | Anexos | Todos os anexos existentes referentes ao problema reportado. Podem ser relatórios, capturas de ecrã, etc. | *Ficheiros*. |
| 8 | Estado | Quando um *bug* é alocado para ser resolvido deve ser conhecido qual o estado em que se encontra. | Os estados possíveis são: New; In Progress; Ready for Test; Done. |
| 9 | Tempo gasto | Após a correção deve ser feito o registo do tempo gasto a replicar o problema, a encontrar a sua origem e a corrigir o mesmo. O tempo gasto deve ser registado em horas. | Por exemplo: 02:20 horas |
| 10 | Criado por | O elemento que procedeu à definição e criação da User Story no Product Backlog. É inserido automaticamente pelo sistema. | *Nome*. |
| 11 | Data de criação | Data de criação. É inserido automaticamente pelo sistema. | |
| 12 | Data de atualização | Data da última atualização. É inserido automaticamente pelo sistema. | |

*Tabela 1 - Template de um Bug*

De forma a trazer clareza ao processo de criação de *bugs* podem ser usadas algumas técnicas para ajudar a trazer robustez ao Bug Backlog, são elas:

- Não introduzir *bugs* duplicados. Antes de proceder à criação de um novo deve ser verificado se esse *bug* já foi reportado.
- Apenas considerar *bugs* encontrados em cenários de produção e aceitação, uma vez que os restantes cenários são questões relativas ao desenvolvimento.
- Completar o mais possível o cartão do bug de forma a fornecer o máximo de informação possível às equipas. Pelo menos uma boa descrição deve ser fornecida.
- Respeitar a classificação da severidade consoante as regras definidas nos processos da empresa.