Universidade do Minho
Escola de Engenharia

André Filipe Bastos Guimarães

Análise, Acompanhamento e Proposta de Melhoria ao Processo de Desenvolvimento do Produto de uma Startup

Março de 2022

Universidade do Minho
Escola de Engenharia

André Filipe Bastos Guimarães

Análise, Acompanhamento e Proposta de Melhoria ao Processo de Desenvolvimento do Produto de uma Startup

Dissertação de Mestrado

Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação de

Professor Doutor Pedro Miguel Gonzalez Abreu Ribeiro

Professor Doutor Paulo Francisco S. Cardoso

Março de 2022

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

AGRADECIMENTOS

A finalização desta dissertação é o culminar do esforço, vontade, suor e ambição aplicados nesta etapa tão importante da minha vida. O término desta fase tão significativa deixa-me extremamente orgulhoso e grato pela maravilhosa experiência que foi frequentar a Universidade do Minho e a oportunidade de ter conhecido as pessoas com quem partilhei momentos memoráveis que marcaram a minha vida académica. Deste modo, não me posso "despedir" sem agradecer a algumas das pessoas que mais apoiaram este caminho trilhado, que foram incansáveis e nunca desistiram de mim.

Em primeiro lugar, agradeço ao meu orientador, o Professor Doutor Pedro Ribeiro, pelo empenho, pelas horas dedicadas em chamadas, e por todo o conhecimento partilhado ao longo destes anos. O meu agradecimento pela confiança depositada em mim ao longo deste extenso processo.

Um agradecimento especial ao meu caro amigo Ricardo Carvalho, fundador da *We Can Charge* e aos seus *co-workers*, Isabel Couto, Rui Couto e Xavier Rodrigues, por me terem recebido de braços abertos e me deixarem aprender e absorver conhecimento de quem já anda nisto há algum tempo. Sempre prestáveis e disponíveis a ajudar, quando assim foi necessário.

Agradeço também a duas pessoas muitos importantes, ao Pedro Cruz Mendes e Sílvia Gonçalves, pela resiliência e determinação em não me deixarem desistir deste desafio a que me comprometi a terminar. Estiveram todo o tempo na minha cabeça porque promessas não existem para serem quebradas.

À minha compincha, Ângela Basto, que me disse todos os dias que possivelmente estava a ceder, "Tu és capaz" (sempre com um sorriso na cara). Isto transmitiu-me a energia e a força, para continuar a escrever, quando as mãos já desistiam.

Aos meus pais, que me apoiaram sempre para terminar o meu curso, sem impor pressão, o que teve um papel preponderante para o terminar na altura que considerei a certa para o fazer.

Por último, aos meus melhores amigos, que em tom de brincadeira, fizeram apostas ao longo dos anos em como eu nunca terminaria a minha dissertação. Fizeram-me nunca perder o foco e, por fim, mostrar que nunca deveriam ter apostado contra mim.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Durante as últimas décadas a engenharia de software tem estado em constante evolução, a um ritmo absolutamente estonteante. Hoje em dia, pensar o mundo sem software seria algo inconcebível, uma vez que dependemos diariamente destes sistemas. A partir de meados do século XX começou-se a aplicar uma das práticas mais importantes na engenharia de software, a gestão de projetos. Esta é uma prática que cada vez mais está presente nas pequenas empresas, através do uso das metodologias ágeis. A indústria surge com a clara pressão que as organizações reduzam custos, melhorem a qualidade dos seus produtos, e que estes sejam lançados o mais rapidamente possível para o mercado ou arriscam-se a perder o comboio da tecnologia. Muitas vezes as pequenas empresas não conseguem acompanhar este ritmo. Deste modo, surge a necessidade, por parte das equipas, de acompanhar e medir o progresso dos projetos, através do uso de métricas.

Esta dissertação tem como objetivo acompanhar, avaliar e apresentar melhorias ao processo de desenvolvimento de uma Startup, através da utilização de uma *framework* de métricas. Para este estudo de caso, realizou-se um acompanhamento contínuo do projeto da Startup, *We Can Charge*, sediada no edifício do Gnration, em Braga, ao longo de 6 ciclos de desenvolvimento do projeto em mãos. Após o acompanhamento dos ciclos de desenvolvimento do projeto, foi realizada uma análise pormenorizada através do uso de uma *framework* de métricas direcionada para o processo de desenvolvimento e apresentado um conjunto de melhorias a esse mesmo processo. A *We Can Charge* utiliza a metodologia Scrum - uma metodologia ágil -, e para monitorizar e gerir o projeto utiliza uma ferramenta designada JIRA Software.

Para cumprir os objetivos propostos nesta dissertação foi necessária a adoção de uma abordagem metodológica. Deste modo, foi selecionado para esta dissertação o estudo de caso, que resultou no contributo direto de um estudo prático, num contexto real.

Os principais resultados obtidos durante a realização desta dissertação centram-se na proposta de melhorias apresentadas ao processo de desenvolvimento da *We Can Charge* e as melhorias positivas exibidas nos ciclos seguintes de desenvolvimento da Startup.

Palavras-Chave

Gestão de Projetos, Scrum, JIRA Software, Framework de Métricas, Startup

ABSTRACT

During last decades, software engineering has been in constant evolution, at an absolutely dizzying pace. Nowadays, thinking about the world without software would be something inconceivable since we depend on these systems on a daily basis. From the mid-twentieth century onwards, one of the most important practices in software engineering began to be applied, the project management. This is a practice that is increasingly found in small companies through the use of agile methodologies. The industry emerges with the clear pressure that organizations reduce costs, improve the quality of their products and that they are launched as quickly as possible to the market, or face the risk of losing the technology train. Small businesses are often unable to keep up with this pace. Thus, a need arises, on the part of teams, to monitor and measure the progress of projects, through the use of metrics.

This dissertation aims to monitor, evaluate, and present improvements to the process of developing in a Startup, through the use of a metrics framework. For this case study, the Startup, *We Can Charge*, headquartered in the Gnration's building, in Braga, was continuously monitored for over 6 cycles of development of the project. After monitoring the project's development cycles, a detailed analysis was carried out using a metrics framework directed to the development process and presented a set of improvements to that same process. *We Can Charge* uses the Scrum methodology - an agile methodology - and to monitor and manage the project uses a tool called JIRA Software.

To fulfill the objectives proposed in this dissertation it was necessary to adopt a methodological approach. Thus, the case study was selected for this dissertation, which resulted in the direct contribution of a practical study, in a real context.

The main results obtained by the completion of this dissertation fall on the proposed improvements presented to the development process of *We Can Charge* and the positive improvements exhibited to the following Startup development cycles.

Keywords

Project Management, Scrum, JIRA Software, Metrics Framework, Startup

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

Lista de Abreviaturas, Siglas e Acrónimos

AC - *Actual Cost*

APMBoK - *APM Body of Knowledge*

BAC - *Budget at Completion*

CPI - *Cost Performance Index*

CV - *Cost Variance*

DFC - Diagrama de Fluxo Cumulativo

EV - *Earned Value*

EVM - *Earned Value Management*

XP - *Extreming Programing*

FDD - *Feature-Driven Development*

ISO – *International Organization for Standardization*

JIT - Just-In-Time

KPI - *Key Performance Indicators*

PV - *Planned Value*

PO - *Product Owner*

PMBoK - *Project Management Body of Knowledge*

SPI - *Schedule Performance Index*

SV - *Schedule Variance*

SDLC – *Software Development Life Cycle*

SQuaRE - *Software Quality Requirements and Evaluation*

WIP - *Work In Progress*

1. INTRODUÇÃO

1.1 Enquadramento

Nos últimos 50 anos, a engenharia de software tem evoluído a um ritmo absolutamente estonteante. Nos dias de hoje, é inconcebível pensar como seria o mundo sem sistemas de software. Sem software não seria possível para o ser humano navegar na internet, divertir-se com jogos de computador ou consolas, usar sistemas operativos ou até mesmo imaginar a possibilidade de explorar o espaço. Na verdade, estes são alguns dos eventos mais importantes na história da humanidade. Engenharia de software é uma disciplina essencial para grande parte das instituições. Mais de 75% da população mundial tem um sistema controlado por software, como o telemóvel, e em 2016 a grande parte destes dispositivos estavam conectados pela internet. Com isto, Ian Sommerville (2006), descreve a engenharia de software como a área que se ocupa de todos os aspetos da produção de software, desde a conceção até à manutenção.

A gestão de projetos é uma das práticas envolvidas na engenharia de software. Esta é uma prática que se modernizou apenas a partir de meados do século XX, começando a ser aplicada em prol do crescimento das instituições. Segundo o APMBOK (*APM Body of Knowledge*) (APM BoK, 2019), gestão de projetos é a aplicação de processos, métodos, conhecimento e experiência com o objetivo de atingir certas metas no projeto, em acordo com os critérios e parâmetros estabelecidos. Na gestão de projetos existem duas abordagens distintas, inseridas na área de engenharia de software, que são largamente usadas pelas organizações: as metodologias tradicionais (Waterfall) e as metodologias ágeis (Scrum, XP - *Extreming Programing*, Kanban).

Durante a realização do estudo bibliográfico sobre as várias metodologias envolvidas em gestão de projetos, a que obteve maior foco foi o Scrum, uma vez que é a metodologia usada no projeto da Startup, em que a dissertação se foca.

Dependendo das características do projeto como a dimensão do projeto/organização e dos *Stakeholders* envolvidos, existe uma escolha natural para o tipo de metodologia a adotar. Normalmente uma organização opta por uma abordagem tradicional quando o produto é previamente especificado, o resultado é previsível e é desenvolvido segundo um exaustivo planeamento. Por outro lado, as organizações optam por uma abordagem ágil, em ambientes com equipas pequenas de trabalho,

preparadas para rápidas mudanças e feedback constante num acordo entre equipa/cliente (Moniruzzaman *et al.*, 2013).

Na indústria, surgiu uma clara pressão por parte dos mercados, para que as organizações reduzam custo, melhorem a qualidade e reduzam o tempo do ciclo de vida de desenvolvimento do produto, para que as empresas se mantenham competitivas. Este é um dos grandes focos das empresas no setor da engenharia de software (Willaert *et al.*, 1998). Deste modo, surge uma clara necessidade das empresas procurarem uma resposta para as exigências impostas pelo mercado, com o objetivo de inovarem e continuarem competitivas.

Desta forma surge a necessidade de encontrar formas de medir o progresso realizado pela equipa, num projeto específico. As métricas são as ferramentas mais utilizadas para medição. Métricas são medidas ou indicadores de avaliação quantificáveis, que são por norma utilizadas para avaliar, comparar e rastrear o desempenho ou produção de um produto (Young, 2020). Por seu lado, métricas de software são medidas usadas para compreender, controlar e melhorar o que foi feito e como é feito o desenvolvimento de um produto. Com o uso de métricas em projetos torna-se possível:

- motivar as pessoas envolvidas e melhorar as práticas usadas no processo;
- identificar ou evitar problemas no processo ou *workflow*;
- prevenir a entrega de produto com defeitos ao cliente final e analisar a atual qualidade do produto; avaliar a qualidade do produto depois da entrega ao cliente final;
- realizar mudanças necessárias no processo ou nas ferramentas usadas.

As métricas auxiliam o processo com o objetivo de estimar o esforço, cumprir/melhorar os prazos, controlar a qualidade, avaliar os custos e produtividade de um projeto (Calazans & Alvarenga, 2014; Canedo & Da Costa, 2018; Kupiainen *et al.*, 2014).

Para esta dissertação foi selecionada uma *framework* de métricas para projetos ágeis, com o objetivo de acompanhar e avaliar o desenvolvimento de um projeto. Com esta investigação, foi possível acompanhar o projeto de uma equipa, inserida num contexto real: uma Startup ligada ao ramo da inovação e da tecnologia.

A Startup que decidiu aceitar a proposta de ser acompanhada num dos seus projetos chama-se *We Can Charge*, pertence à comunidade da Startup Braga e está sediada no edifício do Gnration, em Braga. Este projeto tem o objetivo de criar uma rede de carregadores de carros elétricos, descentralizado das grandes multinacionais, com o objetivo de dar poder a qualquer pessoa ou

empresa em construir a sua rede ou ter o seu carregador. Assim, estas pessoas/empresas podem apostar na autoprodução de energia colocando postos de carregamentos nos seus parques privados.

No âmbito desta dissertação é realizado um acompanhamento do processo de desenvolvimento do produto, através da análise de *Sprints*, da ferramenta de gestão de projetos JIRA Software e das reuniões de equipa semanais. No seguimento da análise realizada, são propostas melhorias ao processo de desenvolvimento da equipa, com a ajuda de uma *framework* de métricas e são visualizados os efeitos diretos da aplicação da proposta de melhoria.

1.2 Motivação e Objetivos

O objetivo principal para esta dissertação é acompanhar, analisar e apresentar uma proposta de melhorias ao processo de desenvolvimento do produto de uma *Startup*, com a utilização de uma *framework* de métricas para projetos ágeis.

Nesta dissertação foram definidos os seguintes objetivos secundários para a conclusão da mesma com sucesso:

- Selecionar uma Startup que esteja disposta a que um dos seus projetos possa ser acompanhado para fins académicos;
- Estudo das metodologias relacionadas com a gestão de projetos;
- Estudo das métricas utilizadas em processos de desenvolvimento de software;
- Estudo de *frameworks* de métricas para projetos ágeis, previamente desenvolvidas;
- Aplicação da *framework* de métricas num caso de estudo em contexto real, num projeto em desenvolvimento;
- Acompanhamento e análise do processo de desenvolvimento do projeto da *Startup*, através da utilização da *framework* de métricas selecionada;
- Apresentar melhorias ao processo de desenvolvimento do projeto em estudo.

1.3 Abordagem Metodológica

Para o trabalho desenvolvido no âmbito da dissertação, existe a necessidade de adoção de uma abordagem metodológica para cumprir os objetivos descritos anteriormente. Para esta dissertação foi adotado o estudo de caso, com o propósito de analisar, acompanhar e apresentar melhorias ao

processo de desenvolvimento de um projeto, na área de desenvolvimento de software. Conforme descrito anteriormente, a dissertação vai acompanhar um caso real, de uma *Startup* ligada ao desenvolvimento tecnológico.

A abordagem metodológica será iniciada por uma revisão literária do estudo de caso, de forma a compreender e aplicar a metodologia corretamente para o bom desenvolvimento da dissertação.

1.3.1 Estudo de Caso

O estudo de caso, segundo Yin (2002), é uma estratégia de pesquisa usada em variadas áreas de investigação, tais como ciência política, psicologia, sociologia, pesquisa de planeamento regional e municipal e organizações, esta última sendo a de maior interesse para a dissertação em desenvolvimento. Esta abordagem é bastante frutífera para desenvolver teorias, avaliar programas e definir intervenções num caso particular devido essencialmente à sua flexibilidade e rigor (Baxter Pamela & Jack, 1990).

Yin (2003), define o estudo de caso como sendo:

> *"An empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident."*

Segundo Yin (2002), o estudo de caso deve ser considerado quando:

- O foco do estudo seja responder às perguntas "Como" e "Porquê";
- Não é possível manipular o comportamento dos envolvidos no estudo;
- Há possibilidade de acompanhar as condições contextuais do objeto em estudo, já que são relevantes para o fenómeno em estudo;
- Os limites não são claros entre o contexto e o fenómeno.

Uma das fases também importantes do estudo de caso é determinar o "Caso" ou a unidade a analisar. Segundo Miles e Huberman (1994), o "caso" é definido por ser um fenómeno, de qualquer tipo, que ocorre em um contexto limitado. Ou seja, o "caso" é em concreto a unidade a analisar da dissertação (Baxter Pamela & Jack, 1990).

Depois de determinar o "caso" é necessário determinar o tipo de estudo de caso. Yin (2003), categoriza os estudos de caso em três diferentes categorias:

- **Explanatório** – Procura responder a uma questão específica que explica uma possível causa ligada diretamente a um evento da vida real. Ou seja, explica a causa/efeito a partir de uma teoria.
- **Exploratório** – Explora as situações nas quais as intervenções avaliadas não são claras ou os problemas são pouco conhecidos. Prepara o terreno para futuro trabalho de investigação.
- **Descritivo** – Descreve um fenómeno ou intervenção e em que contexto da vida real é que ocorre.

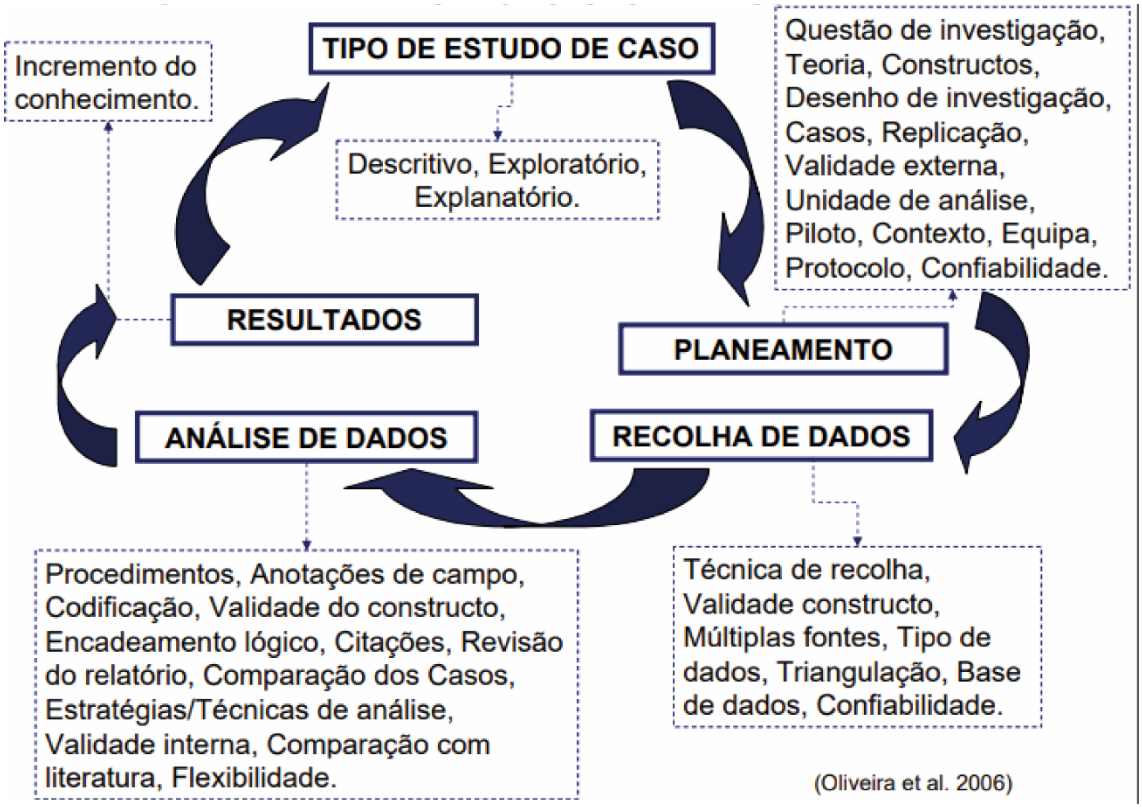

Figura 1 - Framework do Estudo de Caso
(Drebes Pedron, 2008)

O passo seguinte será perceber as várias fases do estudo de caso, para um projeto de dissertação, na área de Sistemas de Informação. Segundo Yin (1998), o estudo de caso passa por três diferentes fases (ilustrado na Figura 1):

- **Planeamento** – Esta é a fase onde se definem questões importantes sobre a investigação proposta. Selecionar o objeto em estudo, as questões a ser respondidas com esta investigação. Ou, conforme referido anteriormente, selecionar o tipo de caso de estudo.
- **Recolha de Dados** – Esta é a fase onde se recolhe a informação para cumprir os objetivos propostos na dissertação. A informação pode ser recolhida de variadas formas e métodos, tais

como: documentação; registos de arquivos; observação direta; observação participante; artefactos físicos e entrevistas.

- **Análise de Dados** – Esta fase consiste em analisar, categorizar, classificar em tabelas ou voltar a verificar as evidências em estudo, tendo em conta as preposições iniciais de um estudo de caso.

Completas estas três fases do estudo de caso, é necessária a redação dos resultados (relatório de dissertação), onde será possível retirar as conclusões do trabalho, que têm o objetivo de contribuir diretamente na comunidade científica (Paré, 2004).

1.3.2 Abordagem Adotada

O estudo de caso foi a abordagem metodológica adotada para esta dissertação. O desenvolvimento desta investigação teve em consideração os parâmetros, fases e tipos de estudos de caso descritos nos pontos acima. Deste modo, para uma melhor compreensão da aplicação desta metodologia, serão descritas as fases de desenvolvimento da dissertação de acordo com a metodologia selecionada e as suas respetivas recomendações.

Para este estudo de caso, o "caso" da dissertação será realizar o acompanhamento do processo de desenvolvimento do projeto da *Startup*, *We can Charge*.

"Como" e "Porquê" será feito? Respondendo à primeira questão, "Como?", o desenvolvimento da dissertação passa por realizar a análise do processo de desenvolvimento do projeto da *We Can Charge*, através do acompanhamento dos ciclos de desenvolvimento Scrum do projeto, com auxílio da utilização de uma *framework* de métricas direcionada para projetos ágeis. Relativamente à questão "Porquê?", esta dissertação tem como objetivo apresentar aperfeiçoamentos ao processo de desenvolvimento da *We Can Charge*, através de uma proposta de melhorias ao processo de desenvolvimento, com auxílio das métricas selecionadas na *framework* utilizada.

Relativamente à recolha de dados, foram selecionadas diferentes técnicas de recolha de informação para o estudo de caso. Observação direta através da participação e observação dos eventos (*Sprint Planning*, *Sprint Review*, *Sprint Retrospective*, *Sprint*) de desenvolvimento do projeto. Documentação e registos de arquivos através da análise dos dados fornecidos pelo JIRA Software, software de acompanhamento e monitorização do projeto *We Can Charge*.

A dissertação envolveu as seguintes etapas:

- **Revisão Bibliográfica** – Estudo das metodologias envolvidas em gestão de projetos, com

maior foco nas metodologias ágeis e estudo das métricas relacionadas com os projetos ágeis. Uma intensa pesquisa foi realizada em relação a *frameworks* de métricas já desenvolvidas previamente em documentos académicos.

- **Recolha de Dados da *We Can Charge*** – Acompanhamento do projeto *We Can Charge* de postos de carregamento elétrico, com foco no processo de desenvolvimento do mesmo. Acompanhamento das reuniões e recolha de dados através do acesso fornecido à ferramenta usada pela equipa para gestão de projetos, o JIRA Software.
- **Análise de Dados da *We Can Charge*** – Análise dos dados recolhidos através da aplicação de uma *framework* de métricas direcionada para projetos ágeis. Analisar as dinâmicas da equipa de desenvolvimento. Utilizar as métricas da *framework* selecionada e com auxílio do JIRA Software, analisar o comportamento do processo de desenvolvimento no decorrer do projeto.
- **Apresentação da Solução e Conclusões** – Apresentação de uma proposta de melhorias ao processo de desenvolvimento do projeto da *We Can Charge*, com observação de resultados imediatos no processo de desenvolvimento do projeto.
- **Escrita da Dissertação** – Envolve a escrita e redação de todas as fases anteriores.

1.4 Estrutura do Documento

Esta dissertação encontra-se dividida estruturalmente em quatro capítulos.

O primeiro capítulo pretende enquadrar o trabalho realizado na dissertação, passando por introduzir o trabalho a desenvolver e apresentar os objetivos da investigação, para cumprir com sucesso a realização da mesma. É também definida e selecionada a abordagem metodológica para este caso em particular, com a finalização do capítulo descrevendo sucintamente a estrutura do documento.

No segundo capítulo é realizada a revisão bibliográfica, onde são descritos os conceitos gerais relativos ao estudo desenvolvido na dissertação. Conceitos na área da Engenharia de Software, Gestão de Projetos, Metodologias Tradicionais e Ágeis, Métricas de Desenvolvimento de Software Ágil e *Framework's* de Métricas para Projetos Ágeis são definidos de forma a prosseguir com o estudo de caso desenvolvido no capítulo seguinte.

No terceiro capítulo é desenvolvido o caso de estudo da dissertação. Aqui é descrito o trabalho realizado na *StartUp*, *We Can Charge*, é traçado o perfil da *StartUp*, são analisados os processos de desenvolvimento do projeto através do acompanhamento dos *Sprints* (ciclos de desenvolvimento) do projeto e é aplicada uma *framework* de métricas de forma obter resultados objetivos sobre a melhoria dos processos de desenvolvimento do projeto.

No quarto e último capítulo da dissertação encontram-se as conclusões obtidas do estudo realizado, tendo em conta os objetivos delineados e os resultados obtidos.

2. REVISÃO BIBLIOGRÁFICA

2.1 Engenharia de Software

Engenharia de Software é a área de engenharia que se foca em desenvolver sistemas de software de alta qualidade ao menor custo. Este conceito surge em 1968 numa conferência com o propósito discutir o que na altura se chamou de "*Software Crisis*" (Sommerville, 2006). Sommerville (2006), define engenharia de software como a engenharia que considera todos os aspetos da produção de software, desde os requisitos de software, definidos numa fase muito embrionária, até aos *updates* necessários realizar, após o sistema estar concebido. Sem a engenharia de software seria impossível alcançar feitos já atingidos pelo ser humano, como ir ao espaço, navegar na internet ou realizar telecomunicações com outras pessoas.

O processo de software é um conjunto de atividades que levam à produção do software. Para simplificar este processo, foram criados diferentes modelos com diferentes perspetivas. Estas metodologias são conhecidas como *Software Development Life Cycle* (SDLC) (Elgabry, 2017).

De acordo com Elgabry (2017), cada processo de software deve incluir as seguintes fases:

- Requisitos do software;
- *Design* e implementação do software:
- Verificação e validação do software;
- Manutenção do software.

Estas são consideradas as principais áreas do processo de software, no entanto, o *Software Engineering Body of Knowlegde* (SWEBoK) (Bourque & Fairley, 2014) define como 15 as principais áreas e subáreas de conhecimento no processo de desenvolvimento. Este é um documento, criado pela *IEEE Computer Society*, com a finalidade de servir como referência na área da engenharia de software (*Software Engineering Course (SWEBOK) | IEEE Computer Society*, n.d.).

Em relação ao produto de software, é possível determinar alguns atributos que refletem a qualidade nestes sistemas. Atributos como a capacidade de manutenção, confiabilidade, eficiência e usabilidade são caraterísticas essenciais de uma boa arquitetura de um sistema de software (Sommerville, 2006).

Em contrapartida, a engenharia de software enfrenta atualmente 3 desafios fundamentais: o desafio da adaptabilidade e heterogeneidade, em que os sistemas se têm de adaptar a diferentes

máquinas com diferentes sistemas; o desafio da entrega, de não comprometer a qualidade dos sistemas, conseguindo encurtar o tempo de entrega, em sistemas com grande complexidade; e o desafio da confiança, essencial para a confiança no software desenvolvido, isto é, é crucial desenvolver técnicas que demonstrem que o software é confiável para os utilizadores que o utilizam (Sommerville, 2006).

2.2 Gestão de Projetos

A gestão de projetos, segundo APMBOK, é a aplicação de processos, métodos, conhecimento e experiência com o objetivo de atingir certas metas no projeto de acordo com os critérios e parâmetros estabelecidos (APM BoK, 2019).

Esta é uma prática que apenas recentemente, em meados do século XX, se modernizou e começou a ser usada para o crescimento e desenvolvimento das organizações (Hoon Kwak, 2003). Contudo, esta é uma prática que tem vindo a ser usada há milhares de anos, desde o Antigo Egipto. Um exemplo disso é a Grande Pirâmide de Gizé (2570 a.C.), uma obra que, devido à sua estrutura e envergadura, ainda na atualidade suscita questões entre arqueólogos sobre o processo de como tal projeto foi terminado. Isto porque no decurso da sua construção, faraós depararam-se com um conjunto de desafios difíceis de superar. No entanto, uma coisa é certa, pouco ou muito, algum tipo planeamento, execução e controlo existiu na altura da sua construção (Hoon Kwak, 2003).

A partir de 1950, com a introdução de novas técnicas e ferramentas, diferentes organizações começaram a usar estas práticas nos seus projetos mais complexos, por exemplo, a marinha usou ferramentas e técnicas de gestão de projetos para desenvolver o projeto Polaris. Durante a década de 60 e 70, organizações como a NASA, o Departamento de Defesa dos Estados Unidos da América e grandes empresas do ramo da construção e engenharia, utilizaram as ferramentas e princípios da gestão de projetos, para gerir os seus custos e prazos nos projetos envolvidos de forma mais eficiente (Hoon Kwak, 2003).

Neste sentido, o objetivo de uma organização aplicar nos seus processos técnicas de gestão de projetos está intrinsecamente relacionado com a necessidade de gerir e controlar eficientemente os recursos disponibilizados para uma atividade específica (Tereso et al., 2019). Estes recursos, conforme ilustrado na Figura 2, estão diretamente ligados com o tempo, custo e rendimento da atividade. Uma última constante que poderá estar envolvida nesta equação é a boa relação com o cliente, visto que de nada vale gerir todos os recursos eficientemente, se a relação com o cliente não for positiva. Daqui se conclui uma visão geral e simplista do mecanismo de gestão de projetos que, atualmente, já inclui outro tipo de características (Kerzner, 2010).

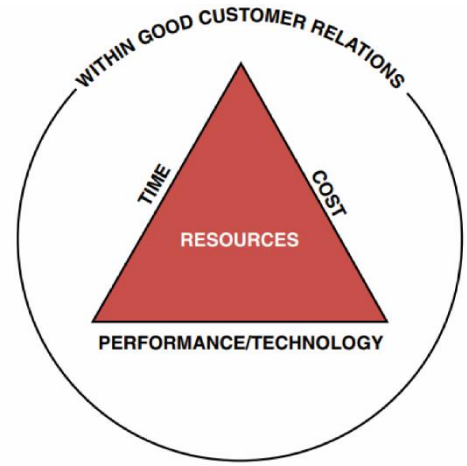

Figura 2 - Visão Geral de Gestão de Projetos
(Kerzner, 2010)

2.3 Metodologias Tradicionais

2.3.1 Introdução

As metodologias tradicionais, também designadas de pesadas ou orientadas à documentação, são aplicadas em situações em que o resultado do seu uso será previsível. Estas metodologias são usadas essencialmente em projetos de grandes dimensões, com requisitos muito bem definidos e com uma estrutura complexa. Ou seja, o produto final resultará em algo expectável, planeado desde o início do projeto. Todo o processo é documentado detalhadamente, de forma evitar quaisquer imprevistos. Isto acontecia devido à falta de ferramentas na altura, como *debuggers,* sabendo que os custos em realizar correções no decorrer dos projetos eram bastante elevados (Soares, 2003). Como se pode perceber a partir da Figura 3, o modelo tradicional deve ser aplicado quando a solução e os objetivos

estão bem definidos. Se o projeto necessitar de maior adaptabilidade, então deve-se selecionar uma outra metodologia (Wysocki, 2014).

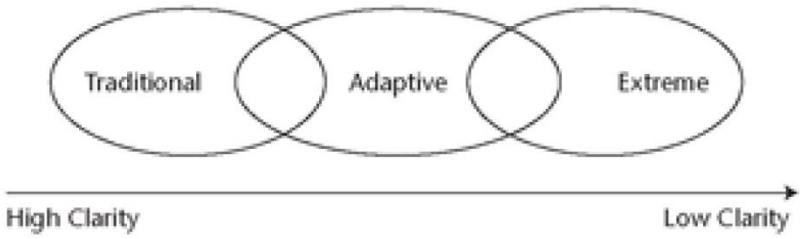

Figura 3 - Abordagens à Gestão de um Projeto
(Wysocki, 2014)

2.3.2 Modelo *Waterfall*

O modelo *Waterfall* é uma das mais antigas metodologias para desenvolvimento de software orientado, desenvolvida por Winston Walker Royce em 1970. Este é um processo sequencial do ciclo de vida de um produto. Segundo Royce, e de acordo com a Figura 4, este é um modelo que passa por diferentes patamares sequencialmente, como se de uma cascata se tratasse, até se obter o produto final (Royce, 1970).

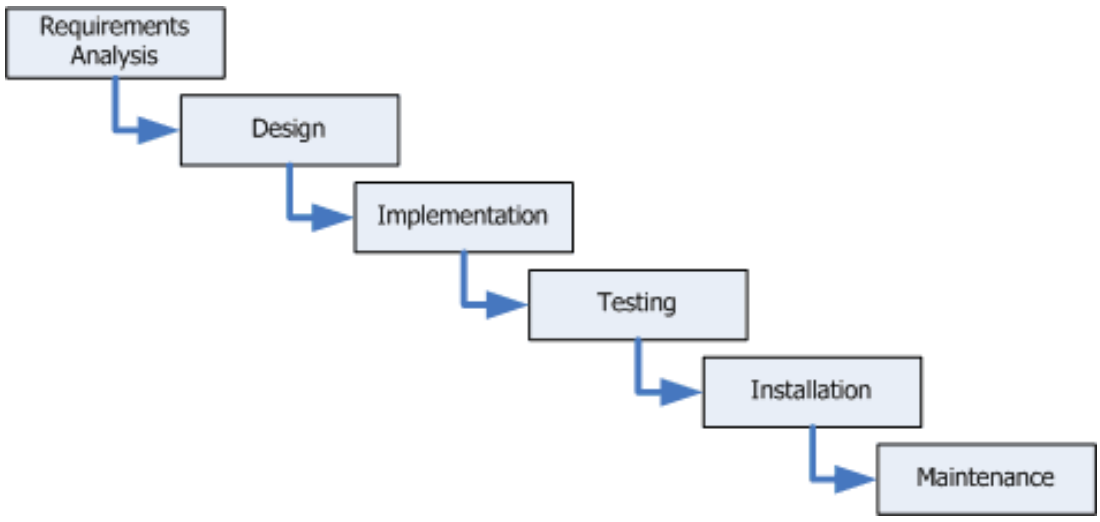

Figura 4 - Ciclo de vida do Modelo Waterfall
Adaptado de: (Royce, 1970)

Royce (1970), defende que para obter sucesso num projeto de desenvolvimento de software é fundamental seguir cinco passos básicos:

- O *design* do programa;

- Documentar detalhadamente todo o processo;
- Repetir o processo, a primeira versão será um protótipo para uma segunda versão do produto final;
- Planear, controlar e monitorizar os testes ao produto;
- Envolver o cliente no processo para obter o melhor produto final.

A partir desta delineação dos princípios, é necessário descrever cada uma das fases do modelo *Waterfall* (Alshamrani & Bahattab, 2015):

- **Requisitos:** Fase em que se procura entender as necessidades do cliente. Com base nesta informação, são definidas as especificações e características de entrada e saída para o produto final. O cliente e os *developers* devem estar em acordo durante esta primeira fase.
- ***Design* do Sistema:** Com a informação recolhida na fase anterior, é possível começar a formalizar uma solução para o *design* do sistema, ou seja, delinear um plano e procurar resolver os problemas que poderão surgir. Encontram-se as soluções mais apropriadas para o *design* do algoritmo, a arquitetura do software, o esquema conceptual da base de dados, o *design* do diagrama lógico e a definição da estrutura de dados.
- **Implementação:** Nesta fase os requisitos delineados para o projeto são convertidos em produto final.
- **Integração e Testes:** Aqui são testadas as soluções desenvolvidas que vão de encontro aos requisitos originalmente delineados. Todos os bugs e problemas encontrados são corrigidos nesta fase.
- **Entrega do Sistema:** Depois de testar os requisitos e características do sistema na última fase, o produto está em condições de ser lançado para o mercado ou entregue ao cliente final.
- **Manutenção:** Depois do produto final lançado, pode existir a necessidade de se fazer algumas alterações, melhorias ou correções de erros encontrados, com o objetivo de ir ao encontro das necessidades dos utilizadores finais.

2.4 Metodologias Ágeis

"Human interactions are complicated and never very crisp and clean in their effects, but they matter more than any other aspect of the work." (DeMarco & Lister, 2014)

2.4.1 Introdução

Atualmente, o ritmo tecnológico atingiu um compasso frenético nunca antes testemunhado na história da humanidade. A sociedade está em constante evolução e, como resultado disso, surgem ferramentas mais capazes de resolver e adaptar os problemas diários que o mundo enfrenta. Os avanços tecnológicos e científicos são notórios diariamente. O mundo encontra-se constantemente e interruptamente conectado e a informação circula a uma velocidade estonteante, com a sociedade em contínua adaptação e mudança. A revolução na área da engenharia modifica a tecnologia, as pessoas, os processos, os serviços e tudo aquilo que está infundido na sociedade atual. Sendo assim, surgiu a necessidade de adaptação para responder e abraçar firmemente, a tempos de mudança. Esta necessidade de mudança deu lugar à criação de metodologias ágeis ou processos ágeis que surgem como uma reação natural aos tradicionais processos de desenvolvimento de software, que os tornavam maçudos, pesados e muito documentados. O produto final não pode esperar pela realização deste massivo processo enquanto a tecnologia e a indústria estão em contante mudança (Cohen *et al.*, 2003).

Highsmith e Cockburn (2001) definem "ágil" como a possibilidade de "entregar rápido, mudar rápido e mudar constantemente", num ambiente de incerteza e turbulência (Alliance, 2017).

A *Agile Alliance* (Alliance, 2017) define desenvolvimento de software ágil como sendo um termo genérico para um conjunto de diferentes métodos e práticas que vão ao encontro dos valores e princípios propostos no Manifesto Ágil. O segredo está na compatibilidade, talento, capacidade e habilidade de comunicação das pessoas envolvidas. Os processos ágeis têm a vantagem de capitalizar as capacidades mais fortes de cada indivíduo e de cada equipa de uma organização (Highsmith & Cockburn, 2001).

Algumas das metodologias ágeis mais conhecidas são o *Extreme Programming* (XP), Scrum, Kanban ou *Feature-Driven Development* (FDD).

2.4.2 Manifesto Ágil

> *"Facilitating change is more effective than attempting to prevent it. Learn to trust in your ability to respond to unpredictable events; it's more important than trusting in your ability to plan for disaster."* (Kent Beck *et al.*, 2001)

O Manifesto Ágil surge da vontade de Dave Thomas, Robert Martin and Jim Highsmith de mudar o rumo sobre a popularidade que, nos anos 90, as metodologias pesadas adquiriram (Kent Beck et al., 2001). Estes 3 indivíduos decidiram realizar um retiro de trabalho com representantes das mais variadas áreas de desenvolvimento de software ágil, e deste encontro resultou o Manifesto Ágil. Apesar da vasta experiência nas mais variadas áreas de desenvolvimento, o grupo representado por 17 pessoas incrivelmente e impressionantemente alcançou um consenso sobre o futuro das metodologias ágeis. Assim, foi possível decretar o que é hoje o Manifesto para desenvolvimento de software ágil, assinado por todos os participantes do encontro, no estado norte-americano de Utah. Este grupo de pensadores livres e independentes, ligados ao desenvolvimento de software, passou a denominar-se como "Aliança Ágil" (Kent Beck *et al.*, 2001).

No Manifesto Ágil, como se pode observar na Figura 5, foram transcritos os seguintes valores (K Beck *et al.*, 2001):

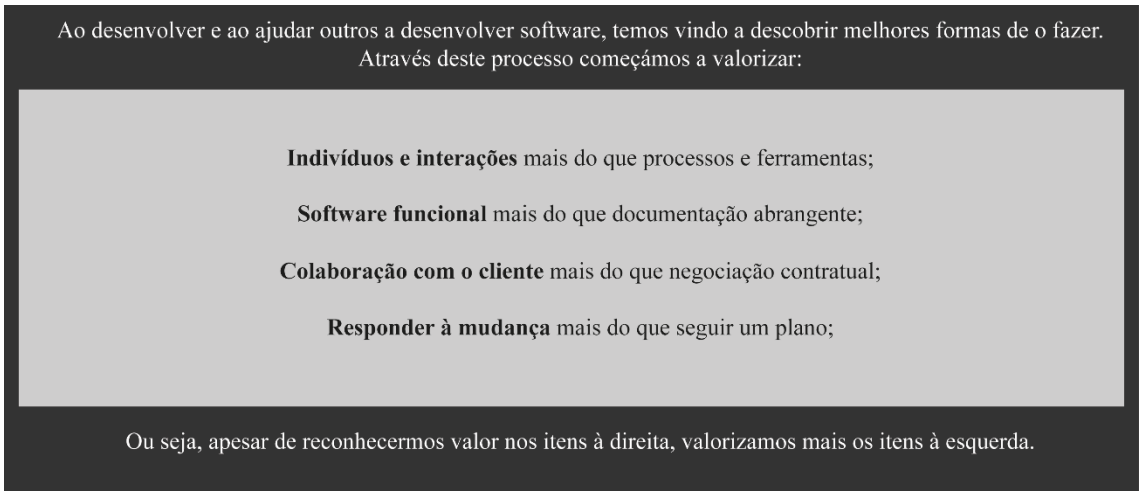

Figura 5- Valores do Manifesto Ágil
Adaptado de: (K Beck *et al.*, 2001)

Para além destes quatro valores fundamentais para a Aliança Ágil, foram também estabelecidos doze princípios básicos. Os princípios apresentados no manifesto ágil estão citados na seguinte Tabela 1 (K Beck et al., 2001):

Tabela 1 - Os doze Princípios do Manifesto Ágil
Adaptado de: (K Beck *et al.*, 2001)

1	A nossa maior prioridade é, desde as primeiras etapas do projeto, satisfazer o cliente através da entrega rápida e contínua de software com valor.
2	Aceitar alterações de requisitos, mesmo numa fase tardia do ciclo de desenvolvimento. Os processos ágeis potenciam a mudança em benefício da vantagem competitiva do cliente.
3	Fornecer frequentemente software funcional. Os períodos de entrega devem ser de poucas semanas a poucos meses, dando preferência a períodos mais curtos.
4	O cliente e *developers* devem trabalhar juntos, diariamente, durante o decorrer do projeto.
5	Desenvolver projetos com indivíduos motivados, dando-lhes o ambiente e o apoio de que necessitam, com confiança de que vão cumprir o seu trabalho.
6	O método mais eficiente e eficaz de passar informação para e dentro de uma equipa de desenvolvimento é através de conversas frente a frente.
7	A principal medida de progresso é a entrega de software funcional.
8	Os processos ágeis promovem o desenvolvimento sustentável. Os *Stakeholders*, a equipa e os *users* deverão ser capazes de manter, indefinidamente, um ritmo constante.
9	A atenção permanente à excelência técnica e um bom *design* aumentam a agilidade.
10	Simplicidade – a arte de maximizar a quantidade de trabalho que não é feito – é essencial.
11	As melhores arquiteturas, requisitos e *designs* surgem de equipas auto-organizadas.
12	A equipa reflete regularmente sobre o modo de se tornar mais eficaz, fazendo os ajustes e adaptações necessárias.

O manifesto ágil resume-se, portanto, a um conjunto de valores e princípios que este grupo de pessoas definiu em consenso com vista a ajudar as equipas de desenvolvimento ágil a entregar ao cliente um produto com qualidade, de forma mais eficiente e rápida (Kent Beck et al., 2001).

2.4.3 Scrum

"Scrum is like your mother-in-law, it points out ALL your faults." Ken Schwaber (1997)

O Scrum foi desenvolvido por Jeff Sutherland com base num artigo escrito em 1986 da autoria de Hirotaka Takeuchi e Ikujiro Nonaka (1986). O artigo descrevia o processo de desenvolvimento de produtos em empresas como a Honda e a Canon.

Scrum, surge a partir de um famoso movimento que acontece no Rugby, em que os jogadores tentam ganhar a bola que se encontra entre as duas equipas, através do trabalho em equipa (Schwaber, 1997). O Scrum é uma abordagem empírica, iterativa e incremental, que ajuda indivíduos, equipas e organizações, a desconstruir problemas complexos, em soluções adaptativas. Segundo o Guia Scrum (Schwaber & Sutherland, 2017), este é um processo que otimiza a previsibilidade e controla o risco, reduz o desperdício, concentra-se no essencial e o conhecimento surge a partir da experiência e das decisões tomadas, com base no que é observado.

Segundo Ken Schwaber e Jeff Sutherland (2017), para aplicar o Scrum com sucesso, deve-se considerar cinco valores fundamentais, como se pode observar na Figura 6. O sucesso da equipa está dependente de valores como coragem, foco, compromisso, respeito e abertura entre os seus elementos.

Figura 6 - Valores do Scrum
(Scrum.org, n.d.)

Os pilares empíricos do Scrum são:

Transparência – Os artefactos devem estar claros, tanto para quem executa o trabalho como para quem os utiliza. Se isto não for aplicado, pode diminuir o valor e aumentar o risco das decisões tomadas.

Inspeção – Os artefactos envolvidos no processo devem ser inspecionados regularmente, para evitar potenciais problemas indesejados e consequentemente atingir os objetivos acordados.

Adaptação – Os artefactos devem ser adaptados e ajustados sempre que assim for necessário.

Conforme referido anteriormente, o Scrum é um processo iterativo, incremental e empírico, em que o ciclo de vida (*Sprint*), como pode ser observado na Figura 7, é composto por eventos com funções diferentes no processo (*Sprint Planning*, *Daily Sprint*, *Sprint Review* e *Sprint Retrospective*) e artefactos (*Product Backlog* e *Sprint Backlog*).

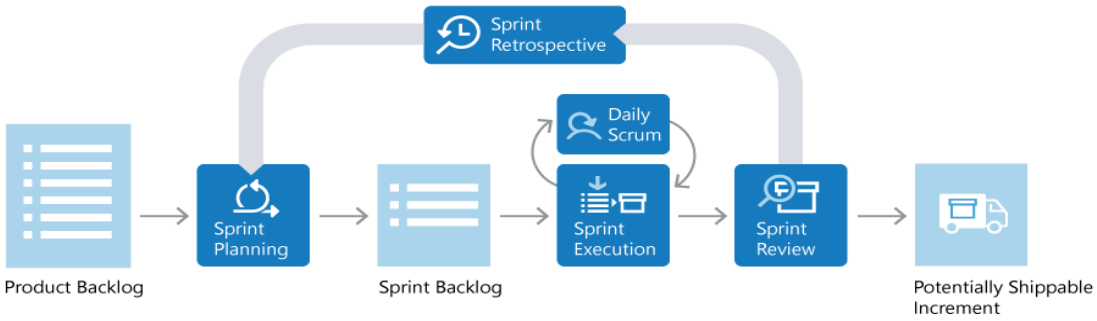

Figura 7 - Ciclo de Vida do Scrum
(Azure DevOps, 2021)

Para além dos eventos e artefactos implícitos nesta metodologia, o Scrum é composto por uma equipa Scrum, normalmente, com três diferentes cargos (*Product Owner*, *Scrum Master* e *Developers*) (Schwaber & Sutherland, 2017).

O Scrum é uma metodologia que ajuda as equipas a gerir o trabalho realizado nos *Sprints*, com o objetivo de melhorar a cada iteração o que será o produto final. Assim sendo, é possível obter maior produtividade a menor custo, melhorar o entrosamento da equipa, melhorar a qualidade e aumentar a satisfação dos *Stakeholders* (Ashbacher, 2010).

Nesta secção é explicado detalhadamente, segundo o Guia Scrum (Schwaber & Sutherland, 2017) e o Livro de Mão Scrum (Sutherland, 2010), cada um dos componentes envolvidos nesta *framework*. Para melhor entendimento é apresentada a Figura 8, ilustrativa dos mecanismos envolvidos.

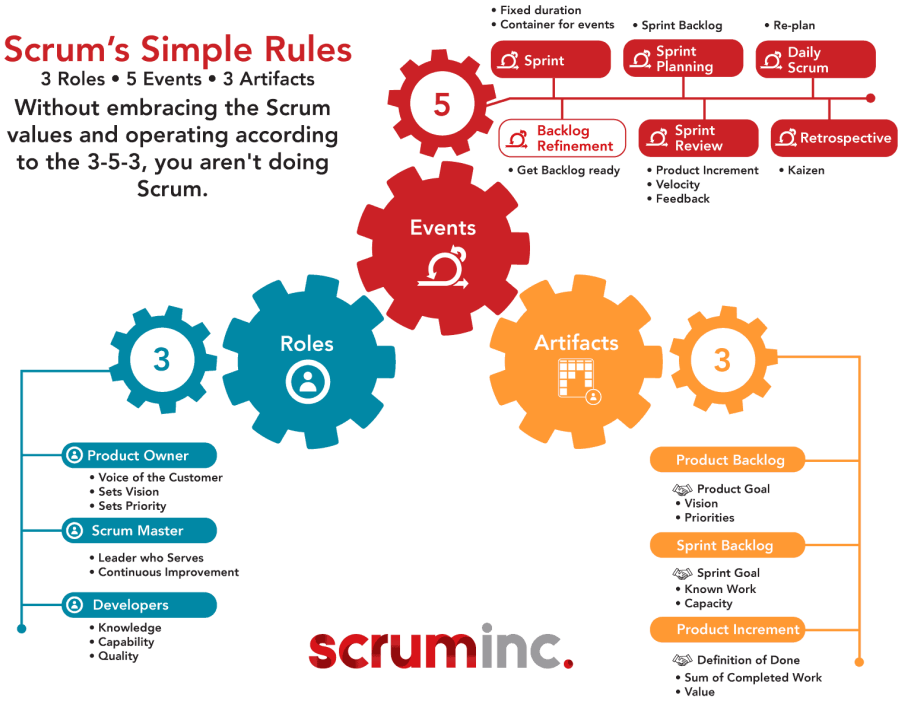

Figura 8 - Estrutura do Scrum
(JJ Sutherland, 2018)

Os artefactos envolvidos na metodologia Scrum são os seguintes:

- ***Product Backlog*** - Este é o primeiro passo a ser dado num projeto. Quando os requisitos estiverem definidos, é necessário prioriza-los e refina-los para uma lista de itens, designada de *Product Backlog*.

  Para cada artefacto existe um compromisso. No *Product Backlog* o compromisso é o *Product Goal*. *Product Goal* é a meta do *Product Backlog*.

- ***Sprint Backlog*** - O *Sprint Backlog* é composto pelo conjunto de itens do *Product Backlog*, que foram selecionados para um *Sprint* específico. Pretende-se seguir o plano estabelecido para incrementar trabalho e passar ao próximo *Sprint*, terminando-o com sucesso e cumprindo o *Sprint Goal da iteração*.

  O *Sprint Goal* é o compromisso do *Sprint Backlog*. Este é objetivo do *Sprint*.

- ***Product Increment*** - O *Product Increment* é a soma de todos os itens da lista do *Product Backlog* completos. Enquanto o *Increment* é a conclusão de um item específico do *Backlog*. Todos os *Increment* devem ser verificados, de forma a funcionarem como um todo.

  Um *Increment* só pode ser dado como concluído quando o objetivo do item for cumprido, ou seja, o estado mudou de "*To Do*", para "*Done*". Este *Increment* tem de cumprir com os padrões de qualidade pretendidos para o produto.

Os papéis definidos para cada elemento dentro da metodologia Scrum são os seguintes:

- ***Scrum Master*** - O *Scrum Master* tem a responsabilidade de assegurar que o projeto segue os princípios, valores e regras definidas pelo Scrum, bem como se o projeto está a progredir conforme o planeado. O *Scrum Master* possui um papel fundamental de contato e interação com o cliente, o gestor e a equipa do projeto. Para além disto, tem a responsabilidade de assegurar que o caminho de desenvolvimento está livre de impedimentos para manter a equipa a trabalhar da forma mais produtiva quanto seja possível.
- ***Product Owner (PO)*** - O PO é o interveniente entre a equipa e os *Stakeholders*. Tem a responsabilidade de maximizar o ROI (retorno do investimento) realizado, através de uma boa interpretação das características do produto em uma eficiente *Backlog List* do produto. Fica também responsável por gerir o *Product Backlog* eficientemente. Definir e comunicar o *Product Goal*, criar e comunicar os itens do *Product Backlog*, bem como priorizar as *features* do *Sprint* e assegurar que elas são claras. As decisões do *PO* devem ser respeitadas.
- ***Developers*** - Estes são os responsáveis por desenvolver, com qualidade, o que será potencialmente o produto final. Trazem toda a experiência e conhecimento para o projeto, para que a cada *Sprint* estejam mais próximos do *Product Goal*. Têm a responsabilidade de criar o *Sprint Backlog* e adaptá-lo sempre que necessário, para atingir o *Sprint Goal*. O número de elementos da equipa varia consoante a complexidade do projeto e cada um é responsável pelo seu próprio trabalho.

Para finalizar a apresentação do Scrum, os eventos envolvidos nesta metodologia ágil são os seguintes:

- ***Sprint*** - O *Sprint* é o principal evento do Scrum, onde "a ação acontece". *Sprints* incrementam trabalho e têm uma duração limitada e fixa de 1 mês (depende de método para método em cada equipa, mas de duração fixa). Sempre que um acaba, outro *Sprint* começa. O *Backlog* é transformado em produto, pelas mãos dos *Developers*, atingindo o *Product Goal*.

  Eventos como o *Sprint Planning*, *Daily Scrums*, *Sprint Review* e *Sprint Retrospective*, fazem parte do *Sprint*. Durante o *Sprint* é essencial atingir o *Sprint Goal*, manter a qualidade, aperfeiçoar o *Backlog* do produto sempre que assim for necessário e reajustar/clarificar o alcance do projeto, em concordância com o *PO*. Algumas das ferramentas mais úteis para acompanhar o progresso de cada *Sprint* são o *Burndown* e *Burnup Chart* ou *Cumulative Flow Diagram* (CFD). Estas métricas ajudam a compreender se o projeto se encontra dentro dos

objetivos definidos.

Um *Sprint* raramente é cancelado, mas pode acontecer. A única pessoa com autoridade de o fazer é o *PO*.

- ***Sprint Planning*** - O *Sprint* inicia com o *Sprint Planning* e tem a duração máxima de 8 horas. O tempo do *Sprint Planning* depende da duração do *Sprint*, se for mais curto, a reunião tem uma duração menor.

  Esta cerimónia, organizada pelo *Scrum Master*, é dividida em duas partes.

  - Numa primeira fase, os *Stakeholders*, *Scrum Team*, *PO* e *Scrum Master* definem os objetivos e funcionalidades, do conjunto de itens do *Product Backlog*, a implementar no *Sprint* que irá iniciar. Os itens do *Product Backlog* selecionados são priorizados consoante o nível de importância. Fica definido quais/qual o *Sprint Goal* que se pretende atingir no *Sprint*.
  - Numa segunda fase, é traçado um plano, entre o *Scrum Master* e a *Scrum Team*, com intuito de decifrar a forma de incrementar com sucesso o produto no *Sprint* que irá iniciar.

Estas tarefas são registadas num documento chamado *Sprint Backlog*, conforme ilustrado na Figura 9.

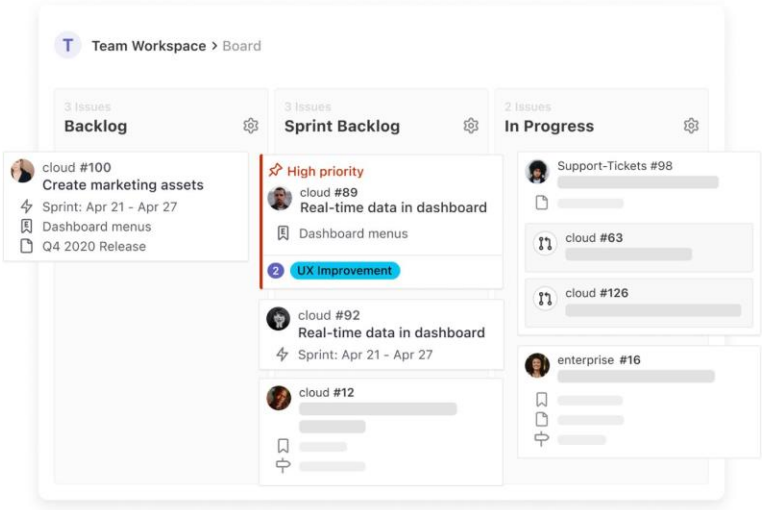

Figura 9 - Exemplo de um Product Backlog
(Mariana Racasan, 2021)

- ***Daily Scrum*** - A *Daily Scrum* é um evento que dura sensivelmente 15 min, com o objetivo de inspecionar se a equipa continua no caminho certo, rumo ao *Sprint Goal*. É essencial que cada

membro reporte o que foi feito desde a última *Daily Scrum*; o que será realizado até à próxima reunião; relatar os impedimentos e obstáculos que cada um pode estar a ter, se assim for necessário. Este evento não tem o intuito de servir para reportar o que quer que seja aos seus superiores, mas sim para partilhar o trabalho feito entre a equipa, evoluindo e fortalecendo os laços dentro da mesma.

Todos os dias úteis do *Sprint* é realizada uma *Daily Scrum*, o que mantem a equipa auto-organizada e focada.

- ***Sprint Review*** - Este é um evento, o penúltimo do *Sprint*, de sensivelmente 4 horas. Aqui traça-se uma retrospetiva do que ficou feito para trás, com o objetivo de inspecionar e adaptar o próximo *Sprint*. A revisão é realizada entre a equipa e o PO.

  Nesta fase, tanto o PO, como os *Stakeholders*, ficam a par do ponto de situação do produto, apresentado pela equipa.

  Se necessário, o *Product Backlog* pode ser ajustado nesta fase, consoante potenciais novas oportunidades de negócio, que poderão surgir.

- ***Sprint Retrospective*** - Neste último evento, que sucede ao *Sprint Review*, é realizada uma reflexão sobre as questões que envolvem o projeto. Como as áreas que podem ser potencialmente melhoradas. A *sprint retrospective* serve também para discutir o que está a funcionar ou a não funcionar e que mudanças podem ser implementadas para o futuro.

2.4.4 Kanban

Kanban surge pela primeira vez na *Toyota System Motors*, pela mão de Taiichi Ohno. O Kanban é criado como suporte ao sistema de produção *Just-in-Time* (JIT) (Kumar & Panneerselvam, 2007). O nome Kanban surge da junção de duas palavras japonesas, *Kan* que significa "sinal" e *Ban* que significa "Quadro". Daí resulta o termo Kanban, quadro para assinalar tarefas (Sugimori et al., 1977).

Esta é uma ferramenta que aplica as práticas do ágil e Lean. O Kanban é um processo visualmente atrativo, em que utiliza cartões chamados Kanban Cards. Cada cartão Kanban representa um item/tarefa do produto, com toda a informação inerente à tarefa. O que facilita a leitura e interpretação da tarefa (Huang & Kusiak, 1996).

O Kanban é um processo de estados. Cada item passa por vários estados até estar completo. Cada cartão move-se do estado *To-Do* (para fazer), até *Done* (completa), passando por *OnGoing* (em progresso). O objetivo é completar todas as tarefas e desenvolver, consequentemente, o produto final. Na Figura 10, podemos ver um exemplo de um quadro Kanban.

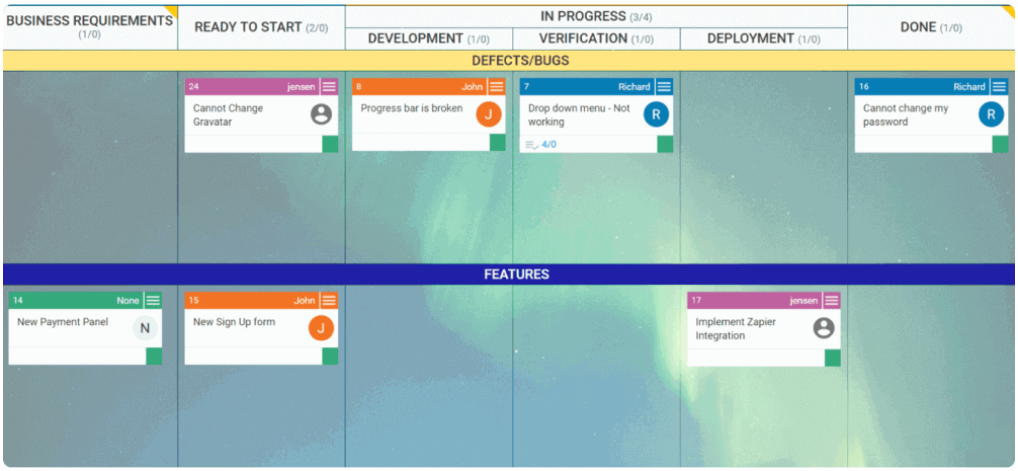

Figura 10 - Exemplo de um quadro Kanban
(Kanbanize, n.d.)

Uma das características principais deste modelo é o limite WIP (*Work-in-Progress*). O Kanban restringe a quantidade de tarefas que podem estar a ser trabalhadas, ao mesmo tempo (definida pela equipa). Este WIP ajuda a equipa a focar-se num grupo restrito de tarefas, o que faz com que o trabalho executado seja fluido e organizado. Deste modo, a equipa resolve eficientemente as tarefas com maior rapidez sem a possível distração do acumular de tarefas (Kanbanize, 2019).

A base fundamental do Kanban estrutura-se por (Kniberg & Skarin, 2010):

- Visualizar o *workflow* do processo;
- Limitar o WIP;
- Estimar o tempo de um item até estar finalizado.

Para gerir da melhor maneira o *workflow* do projeto, o Kanban segue um conjunto de princípios e propriedades que ajudam a organização neste processo. Para o bom funcionamento do Kanban é necessário seguir um conjunto de bons princípios para que as propriedades do Kanban sejam usadas da melhor forma. Sendo assim, David J. Anderson (Kanbanize, n.d.-b), formulou um conjunto de princípios usados no Kanban:

- **Começar com o que se está a fazer agora:** Não realizar mudanças só porque se vai começar a utilizar o Kanban. Essas mudanças podem ser realizadas com o tempo e a experiência, desde que a equipa se sinta confortável.

- **Concordar em adicionar pequenas mudanças:** O Kanban incentiva a pequenas mudanças incrementais, nunca a mudanças radicais.
- **Respeitar os cargos, responsabilidades e papeis atuais:** Como o Kanban não impõe mudanças nas organizações, não é necessário fazer alterações nos atuais cargos e funções da organização. Se a equipa sentir necessidade, colaborativamente, irá implementar as mudanças necessárias.
- **Encorajar atos de liderança a todos os níveis:** Todos os intervenientes na equipa devem -se sentir capazes de adquirir atitudes de liderança. Isto é o que o Kanban encoraja. Ideias surgem de todos os lados e de todas as pessoas.

As propriedades principais que levam ao bom funcionamento do processo Kanban são as seguintes (Ahmad et al., 2018):

- Visualizar o *workflow*;
- Limitar o WIP;
- Medir e gerir o *workflow*;
- Tornar as políticas do processo explícitas;
- Usar modelos para reconhecer oportunidades de melhoria.

2.4.5 Scrumban

"We know Scrum and Kanban as flavors of Agile. Scrum is best-suited for products and development projects. Kanban is best for production support. We use Scrumban – which combines the best features of both – for maintenance projects."

Savita Pahuja (2015)

Quando se tem à disposição duas metodologias *Lean*, como o Scrum e o Kanban, porque não combiná-las e torná-las num processo ideal. Esta foi a ideia de Corey Ladas, quando decidiu escrever o "*Scrumban – Essays on Kanban Systems for Lean Software*". Corey Ladas decidiu então combinar duas metodologias, o Scrum e o Kanban, agregando as melhores práticas e regras de ambos os métodos, criando o Scrumban, com o intuito de unir o melhor de dois mundos. Enquanto por um lado

o Scrum aproveita o fato de esta ser uma *framework* com uma abordagem ágil, por outro lado, o Kanban, aproveita o fato desta metodologia estar constantemente a encorajar o aperfeiçoamento do processo, o que permite à equipa melhorar os seus processos continuamente (Ladas, 2009; Pahuja, 2015).

No Scrum, como se pode observar na Figura 11, o processo tem início com o planeamento do *Sprint Backlog*. As tarefas vão sendo cumpridas e apenas quando o trabalho é todo concluído é que se passa ao planeamento do próximo *Sprint*.

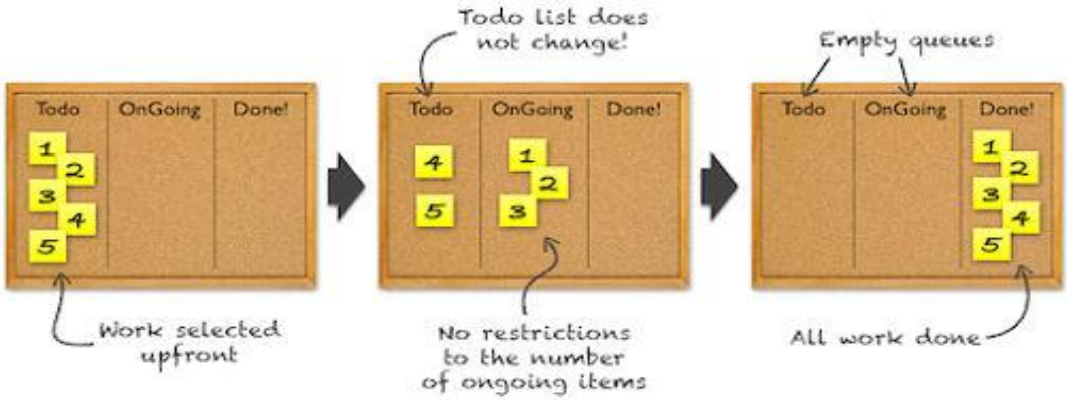

Figura 11 - Processo Exemplificativo do Ciclo Scrum
(Pahuja, 2015)

No Kanban, como se pode perceber pela Figura 12, existe um número limitado de tarefas que podem ser realizadas ao mesmo tempo, chamado de WIP *Limit*. Apesar do número limitado de tarefas em progresso, estas podem ser trocadas com tarefas da lista *To Do*, se assim se achar conveniente. Em Kanban, a terminologia *Sprint* não existe, o trabalho apenas flui naturalmente enquanto existirem tarefas.

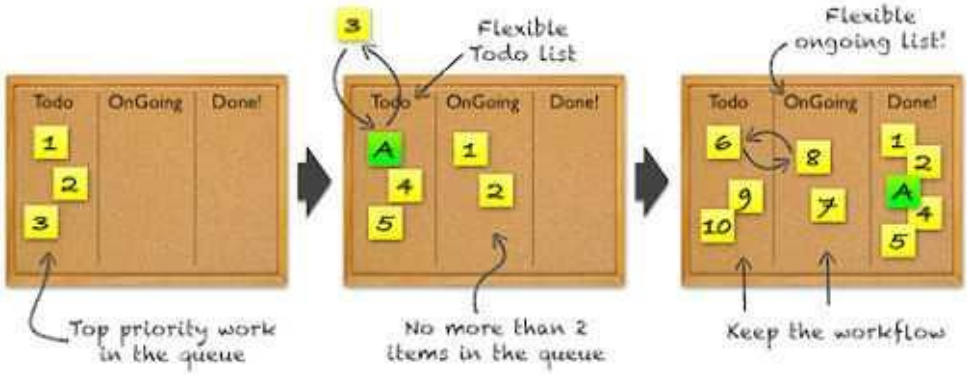

Figura 12 - Processo Exemplificativo do Kanban
(Pahuja, 2015)

Assim sendo, o Scrumban é a combinação das melhores práticas do Scrum, com os melhores princípios do Kanban. O Scrumban é uma aplicação do método Kanban e consequentemente é natural que os princípios básicos sejam aplicados na *framework*. Os princípios básicos são (Yuval Yeret, 2012):

I. **Começar pelo que faço agora.** Definir os cargos e aplicar todos os mecanismos do Scrum, como eventos e artefactos.

II. **Abraçar a mudança.** Tornar o processo adaptativo para ajudar as organizações a serem mais eficientes. O importante é nunca parar de adaptar e evoluir.

III. **Respeitar as funções e responsabilidades do processo.** Começar com a abordagem Scrum e ir adaptando a metodologia para se entrosar nas ideologias da organização. O Scrumban e o Kanban são exatamente sobre isto, considerar e introduzir mudanças, sempre que realmente essas mudanças fazem sentido.

O Scrumban adota algumas das práticas mais importantes do Kanban e Scrum, resultando nos seguintes pontos (Khan, 2014):

I. Visualizar o *workflow*;

II. "Puxar" trabalho;

III. Limitar os itens do WIP;

IV. Tornar as políticas da equipa explícitas;

V. Planear as Reuniões;

VI. Realizar as cerimónias de *Daily*, de *Review* e *Retrospective Sprint*;

VII. Usar as métricas e estimativas do Scrumban.

2.4.6 Métricas de Desenvolvimento de Software Ágil

Métricas, segundo a Investopedia (YOUNG, 2020), são medidas quantificáveis usadas para avaliar, comparar e seguir o desenvolvimento de um produto. Ou seja, enquanto métricas são simplesmente medidas predefinidas, métricas ágeis são medidas predefinidas que ajudam as equipas de trabalho nos projetos, a monitorizar a qualidade e a produtividade, durante as várias fases de desenvolvimento. Medir os níveis de produtividade da equipa, ajuda a manter os níveis de rendimento no projeto (Chowdhury, 2021).

Nos projetos que usam metodologias ágeis são aplicadas 6 diferentes métricas ágeis, que ajudam otimizar o processo ao longo do desenvolvimento do produto. Essas métricas são as seguintes (Hayes et al., 2014; Radigan, n.d.):

- *Sprint Burndown*;

- *Epic and Release Burndown*;
- *Release Burnup*;
- *Velocity Chart*;
- *Control Chart*;
- *Cumulative Flow Diagram*.

Para melhor compreensão de cada uma das métricas ágeis enunciadas, será realizada uma pequena descrição das mesmas (Radigan, n.d.).

Sprint Burndown

O *Sprint Burndown* reporta a quantidade de trabalho completo ao longo de um *Sprint*. Como se pode perceber na Figura 13, no gráfico *Burndown*, o eixo horizontal representa o tempo do *Sprint* e o eixo vertical representa a quantidade que ainda é necessário realizar no *Sprint*, normalmente descrito em *Story Points*.

O objetivo do *Sprint Burndown* é verificar se será terminado o trabalho assumido para o *Sprint*, antes do final do mesmo.

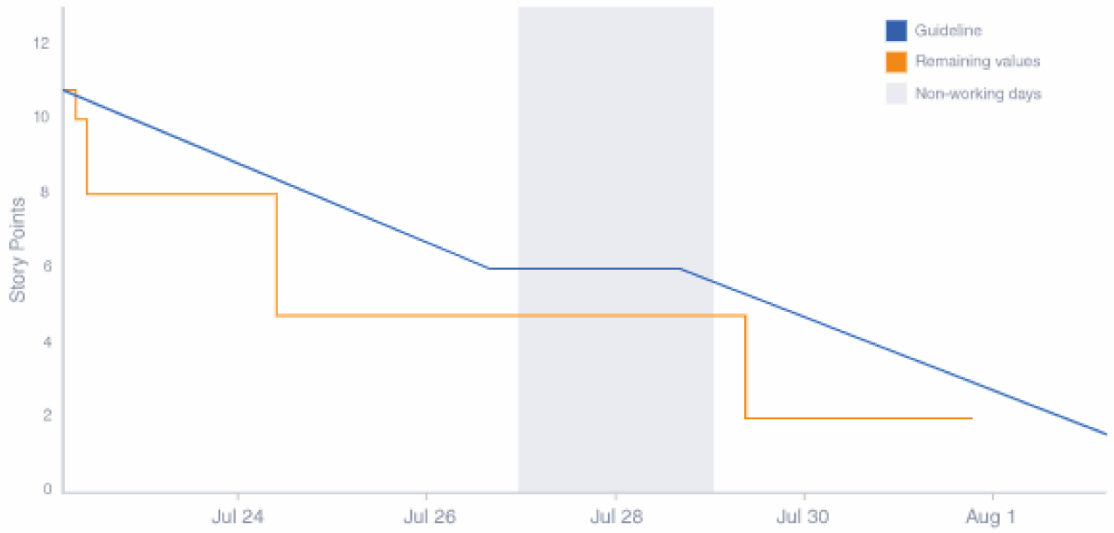

Figura 13 – Exemplo de Gráfico Burndown
(Radigan, n.d.)

Epic and Release Burndown

Epic and Release Burndown combina informação sobre *epics* e *releases* ao longo dos *Sprints* do projeto. Além desta informação é possível perceber o *Scope Creep* (introdução de requisitos novos num projeto previamente definido) de cada *Sprint*. Na Figura 14 é ilustrado um exemplo de um gráfico *epic and release burndown* em que o eixo x representa o *Sprint* e o eixo y são representados os *Story Points* (métrica usada para estimar a dificuldade de uma tarefa para a completar).

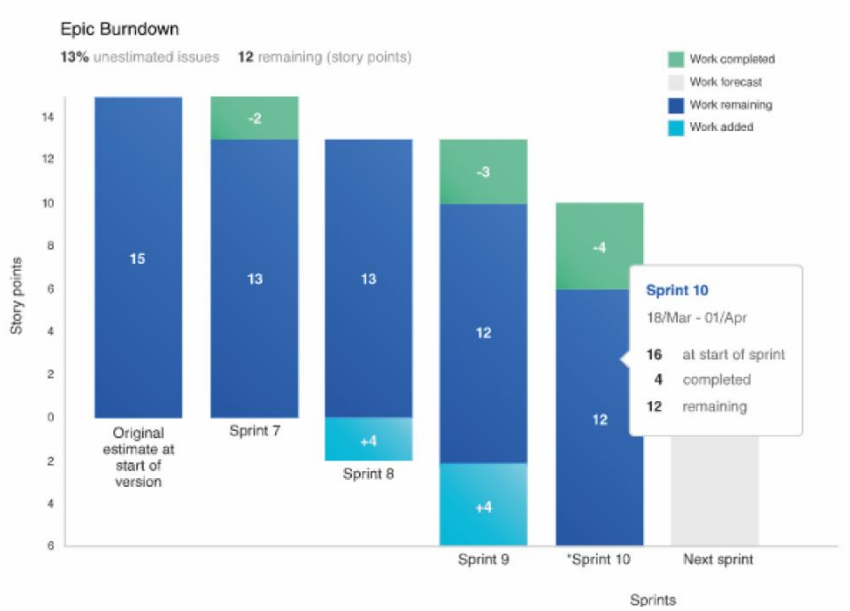

Figura 14 - Exemplo de um Gráfico Epic and Release Burndown
(Radigan, n.d.)

O gráfico tem como objetivo mostrar e manter a equipa a par do trabalho adicionado, do trabalho a realizar, do trabalho completo e do que lhes espera no *Sprint* seguinte.

Release Burnup

O *release burnup* é um gráfico que mostra a quantidade de trabalho completo num *Sprint*, em comparação, com o alcance total do *Sprint*. Ao longo do *Sprint,* o trabalho completo aumenta e o número de tarefas vai diminuindo, com o objetivo de obter sucesso na finalização da totalidade das tarefas.

Na Figura 15, o *release burnup* é representado no eixo x por os dias do *Sprints* e no eixo y pelo número de *Story Points* entregues. A linha (a vermelho) no topo do gráfico representa o total de *Story Points* planeado para o *Sprint*. Uma outra linha (cinzenta) na diagonal serve como referência para atingir o *Sprint Goal*. A linha (verde) representa o total de *Story Points* completos, durante o *Sprint*. A distância entre a linha vermelha e a verde representa o total *Story Points* que ainda são necessárias realizar. Quando as duas linhas se intersetarem, o objetivo do *Sprint* fica completo.

Figura 15 - Exemplo de um Gráfico Burnup
(Atlassian, n.d.)

Velocity

Velocity representa o número de *Story Points* completos por *Sprint*. A cada *Sprint* é possível ajustar as *Story Points* que a equipa Scrum se propõe. Assim, a cada *Sprint* pode aumentar ou diminuir o número de *Story Points* por *Sprint*, dependendo da velocidade que a equipa está a cumprir os objetivos. Cada equipa tem o seu próprio ritmo.

Na Figura 15, o eixo x representa o número de *Sprints* e no eixo y o número de *Story Points* (propostos e completos).

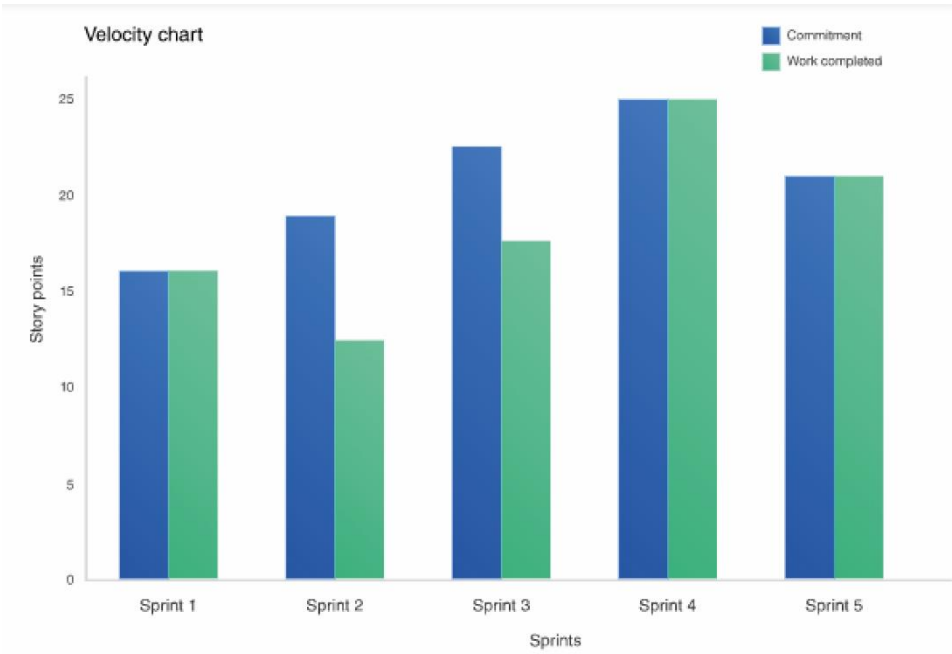

Figura 15 -Exemplo de um Gráfico Velocity
(Radigan, n.d.)

Control Chart

O *control chart* foca-se no tempo total gasto por cada tarefa desde que entra no estado "*In Progress*" para "*Done*".

Equipas Scrum podem beneficiar ao otimizar o tempo por ciclo. Medir o tempo gasto por ciclo é uma boa maneira de melhorar o processo de desenvolvimento da equipa.

O objetivo do *control chart* é encurtar e manter a consistência do *Cycle Time*, independentemente do tipo de trabalho.

Na Figura 16, o eixo x é representado por data de transição de problemas e o eixo y representa o tempo decorrido em dias.

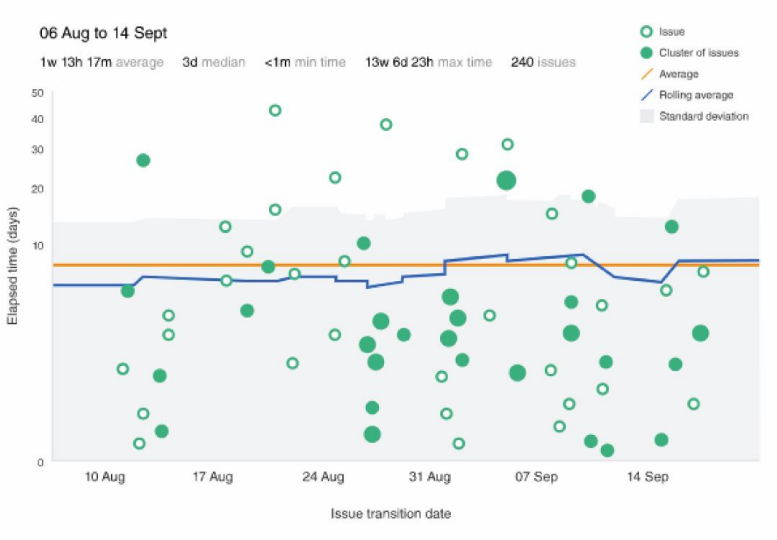

Figura 16 - Exemplo de um Control Chart
(Radigan, n.d.)

Cumulative Flow Diagram

Cumulative Flow Diagram (CFD) é um gráfico que combina a informação de três diferentes parâmetros num período específico do projeto.

I. Tarefas ainda por realizar;

II. Tarefas em progresso;

III. Tarefas completas.

O CFD deve ter um comportamento suave da esquerda para a direita, como se pode observar na Figura 17. O eixo x representa o tempo selecionado de observação e o eixo y o número de tarefas.

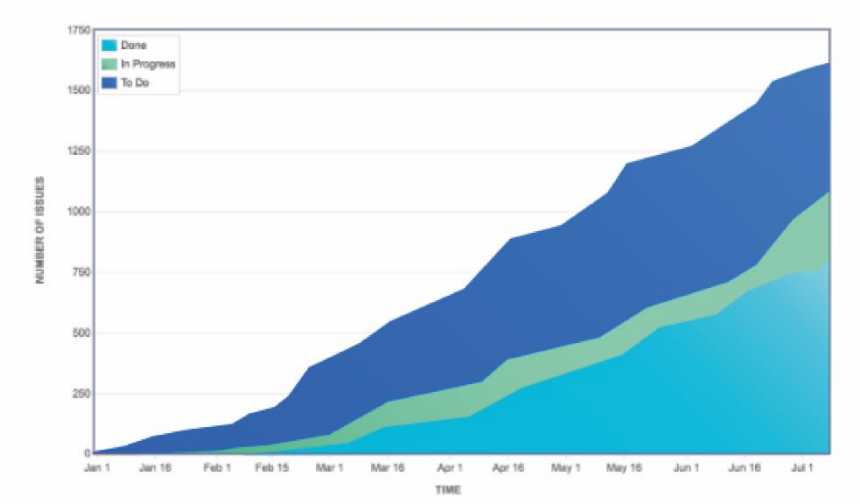

Figura 17 - Exemplo de um Cumulative Flow Diagram
(Radigan, n.d.)

Uma das vantagens do CFD é poder obter outras métricas e a partir daí visualizar e analisar dados do projeto. Através do CFD é possível extrair o WIP, *Throughput* e o *Cycle Time* (mencionado no ponto anterior).

WIP

O WIP é o número de tarefas que a equipa se encontra a desenvolver num momento específico do projeto, ou seja, tarefas que são dadas como iniciadas mas ainda não foram concluídas (Kanbanize, n.d.; Lynn, n.d.). O WIP pode ser caraterizado como uma métrica ou como um limite (como é explicado na subsessão **2.4.4**), dependendo da forma como é usado (Gustafsson et al., 2011).

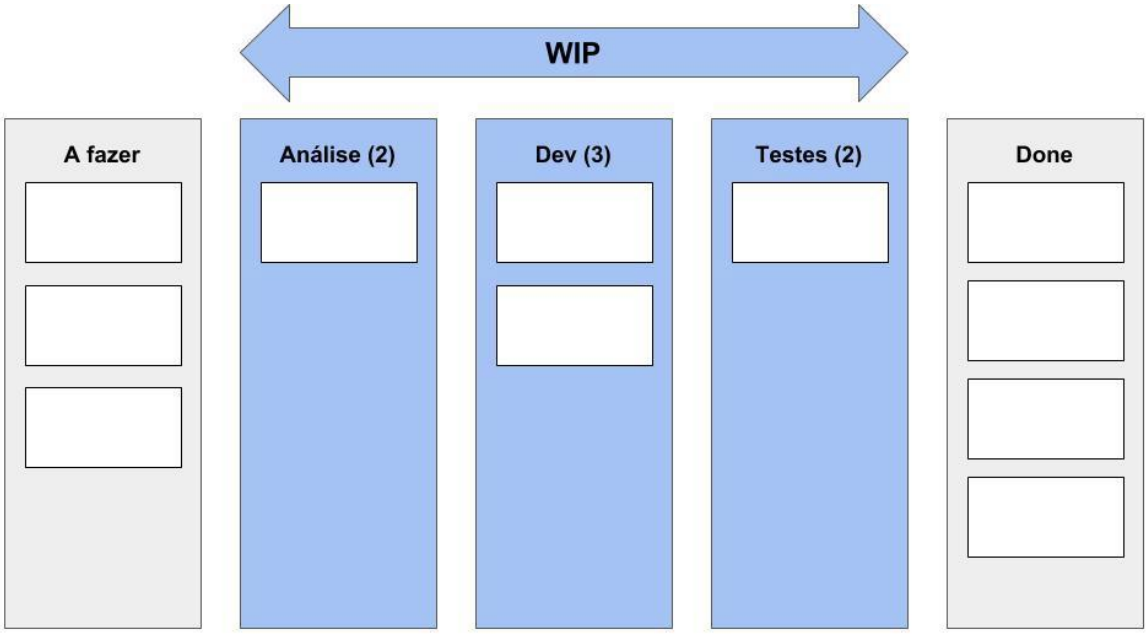

Figura 18 - Exemplo de Utilização do WIP num Projeto
(Raphael Batagini, 2019)

Como métrica, o WIP, tem o objetivo de adaptar o fluxo de tarefas em desenvolvimento, de forma, a melhorar o processo de desenvolvimento do produto. Adaptando o WIP, é ajustada a carga de trabalho e aumenta cooperação entre elementos da equipa. Um WIP baixo indica que os elementos da equipa trabalham em grupo, o contrário pode significar excessivo trabalho e foco reduzido (Gustafsson et al., 2011).

Como limite, o WIP, é uma forma de restringir o número de tarefas em execução, o que previne o acumular de tarefas. Reduz o fluxo de tarefas em desenvolvimento, ajuda a manter o quadro de tarefas limpo e a manter a equipa focada nas tarefas que importam. Sendo assim, o WIP deve estar adaptado à realidade da organização e manter o rendimento do projeto no limite (Gustafsson et al., 2011).

Throughput

Throughput é definido por ser a quantidade de trabalho entregue num período específico de tempo. Esta é uma métrica usada no Kanban que demonstra se o processo da equipa de desenvolvimento tem vindo a ser produtiva ou não (Pavel, 2021).

Uma das formas de aumentar a produtividade do processo é assegurar que o WIP não está acima dos limites do sistema e ao mesmo tempo estarem focados em reduzir o *Cycle Time*. Um princípio básico que explica a relação entre estas variáveis é a *Little's Law.*

$$Throughput = WIP \div Cycle\ Time$$

Esta equação mostra essencialmente que para obter máximo rendimento, o *Cycle Time* deve diminuir continuamente, enquanto o WIP não deve exceder a capacidade de processamento das tarefas.

Earned Value Management

As métricas anteriormente descritas são as usadas geralmente em projetos ágeis. No entanto, não são as únicas. Há outras métricas que normalmente provêm de abordagens tradicionais e que podem ser valiosas para projetos ágeis (Sulaiman et al., 2006). Uma das mais importantes técnicas usadas na gestão do alcance, tempo e custo e que mede o desempenho do projeto é o *Earned Value Management* (EVM). Este é um método que integra métricas de avaliação e integração relacionadas com o alcance, custo e tempo do projeto (Sulaiman et al., 2006).

O conceito de EVM foi introduzido na década de 1890 como uma maneira dos primeiros engenheiros industriais medirem o desempenho das fábricas americanas. Basicamente é definido um

"*cost variance*" relacionando "*earned standards*" com "*actual expenses*" para determinar desempenho nestes casos (Cabri & Griffiths, 2006). Em 1962, o EVM finalmente é reconhecida como uma técnica de gestão de projetos introduzida pela marinha Americana e mais tarde é incluída pelo Instituto de Gestão de Projetos, no famoso guia *Project Management Body of Knowledge* (PMBoK) (Cabri & Griffiths, 2006; Project Management Institute, 2008; Sulaiman et al., 2006).

As métricas fundamentais usadas pelo EVM, retiradas da ISO 21508:2018 (ISO, 2018), podem ser observadas na Tabela 2. Estas métricas servem como base a todas as outras métricas (métricas de desempenho atual ou futuro) que podem ser usadas para medir o desempenho de um projeto (Dash, 2020).

Tabela 2 - Métricas Fundamentais do EVM
(Dash, 2020)

Nome	Descrição
***Budget At Completion* (BAC)**	O somatório de todos os orçamentos estabelecidos para a execução do trabalho
***Planned Value* (PV)**	O orçamento autorizado para o trabalho agendado
***Earned Value* (EV)**	Medida do trabalho executado, expressa em termos do orçamento autorizado para o trabalho
***Actual Cost* (AC)**	Custo realizado incorrido no trabalho executado de uma atividade, durante um período específico

A partir destas métricas fundamentais é possível calcular outras métricas normalmente usadas no modelo EVM. As principais são mencionadas na Tabela 3 seguinte (Dash, 2020).

Tabela 3 – Métricas de Cálculo EVM do Desempenho de um Projeto
(Dash, 2020)

Nome	Formula	Descrição
***Schedule Variance* (SV)**	$SV = EV - PV$	Valor pelo qual o projeto está atrasado ou adiantado em relação à data de entrega planeada
***Cost Variance* (CV)**	$CV = EV - AC$	Quantidade de défice ou excedente orçamentário num determinado momento
Schedule Performance Index	$SPI = \frac{EV}{PV}$	Grau de eficiência com que a equipa do projeto está a realizar o trabalho

(SPI)		
***Cost Performance Index* (CPI)**	$SPI = \frac{EV}{AC}$	Grau de eficiência de custos do trabalho realizado

Como foi dito anteriormente as métricas EVM são geralmente aplicadas em projetos tradicionais. Não quer dizer que não seja possível aplicar estas mesmas métricas em projetos que adotam metodologias ágeis. Nas metodologias ágeis as medições são realizadas no fim de cada *Sprint*, quando se obtém dados concretos sobre o AC e *Velocity* do projeto (Cabri & Griffiths, 2006; Dash, 2020; Sulaiman et al., 2006).

Na Figura 19, pode-se observar a representação das curvas efetuadas por estas métricas, usadas em projetos ágeis (Dash, 2020).

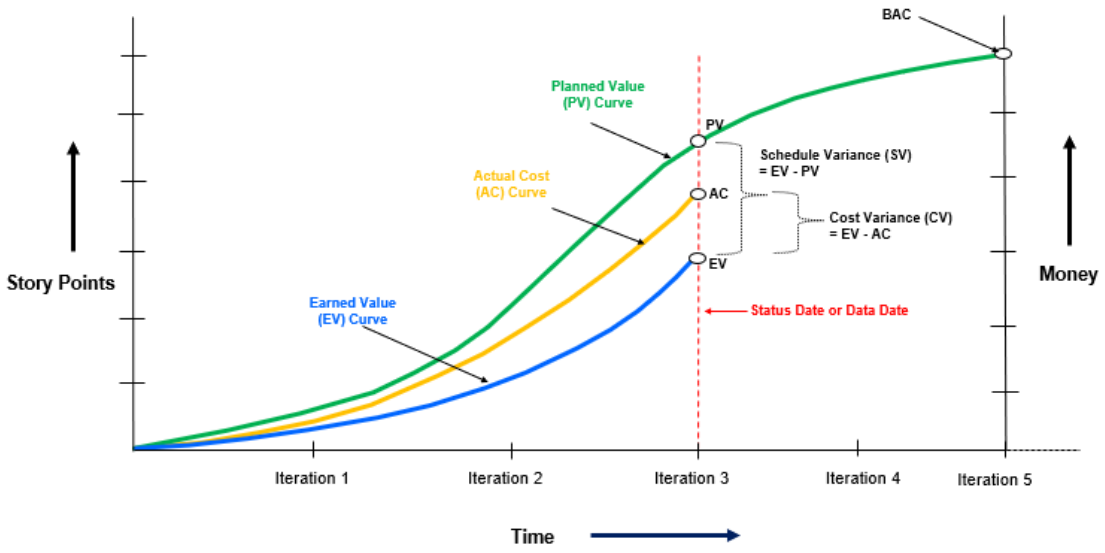

Figura 19 - Exemplo do Comportamento de EV, AC, PV e BAC
(Dash, 2020)

2.5 Framework's de Métricas para Projetos Ágeis

Como foi mencionado no capítulo anterior, métricas são indicadores importantíssimos que ajudam as empresas a recolher uma quantidade considerável de informação sobre os seus projetos. Indicadores podem revelar informações sobre a qualidade do produto, a velocidade com que o projeto tem progredido, estimativas de custos, rendimento do projeto e da equipa, entre outros (Padmini et al., 2015). Esta é uma forma de manter o foco, a qualidade e a produtividade das equipas no desenvolvimento de projetos (Chowdhury, 2021).

Uma das maiores preocupações nas organizações é saber como os seus projetos são conduzidos e se vão atingir os objetivos dentro dos limites estabelecidos. As organizações procuram respostas para perguntas como: "Vamos atingir os resultados que desejamos?", "Vamos atingir os objetivos para o projeto?", "Vamos alcançar a satisfação dos clientes?" e "Vamos alcançar o ROI?". O sucesso de um projeto depende da capacidade de prever e estabelecer/cumprir compromissos em relação aos produtos e serviços desenvolvidos (O'Hara, Susan & Levin, 2000).

Existe um grande contraste em relação à forma como as métricas são aplicadas numa abordagem de desenvolvimento ágil e tradicional (Kupiainen et al., 2015). No entanto, para medir um projeto, deve-se ter em consideração, independentemente da abordagem aplicada, 4 áreas distintas: *Schedule*, *Scope*, *Budget* e *Quality*. A diferença são as métricas a aplicar e a forma como se aplicam nas distintas metodologias. Numa *framework* flexível e iterativa como a ágil, as métricas surgem como uma forma de melhorar o processo a cada iteração, numa abordagem tradicional as métricas focam-se na entrega final do projeto, considerando todos os indicadores iniciais planeados para o projeto (Benavides, n.d.). Deste modo, surgem as KPIs (*Key Performance Indicators*), essenciais para medir o rendimento de qualquer atividade que é importante para o negócio (Geckoboard, 2019).

Com isto, estão reunidas as condições para apresentar as *frameworks* de métricas estudadas para projetos ágeis.

2.5.1 Agile Metric Framework

Esta primeira *framework* foi desenvolvida numa dissertação de 2018, realizada pelo aluno Luís Correia, da Universidade do Minho. Esta *framework*, de acordo com a Figura 20, encontra-se dividida em 5 áreas (Correia, 2018): EVM, Planeamento, Desenvolvimento, Qualidade e *Stakeholders*.

O modelo foi desenhado com o propósito de cada uma das áreas exteriores possa convergir para o centro, para que todas elas estejam conectadas. Cada uma das áreas é responsável por medir diferentes aspetos relacionados com o desenvolvimento do projeto.

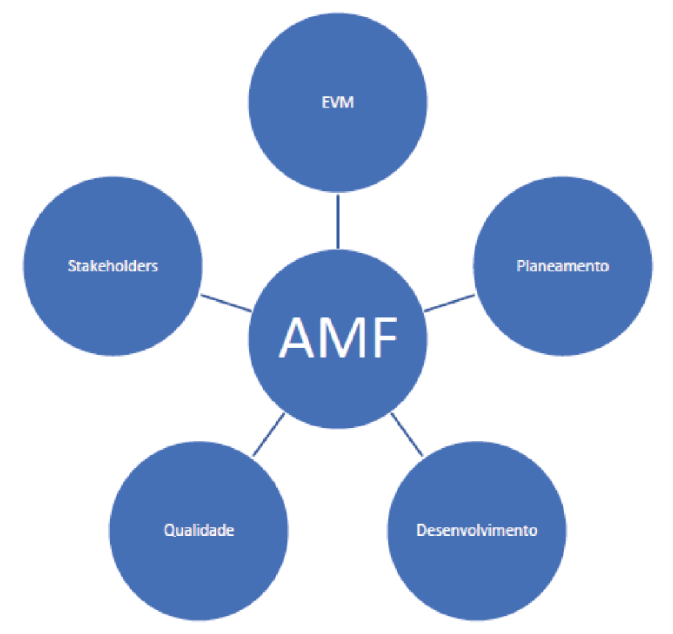

Figura 20 - Estrutura da Agile Metric Framework
(Correia, 2018)

Seguidamente será descrita cada uma das áreas do modelo:

- **EVM** - Esta área é constituída por 8 métricas, 6 calculadas e 2 valores previamente definidos pela equipa. Estas métricas pretendem avaliar o projeto em termos de custo, tempo, e índices de desempenho.
- **Planeamento** - Esta área é responsável por compreender se o planeamento estipulado, para cada fase do projeto, está a ser cumprido. Serve como suporte ao desenvolvimento e apoio ao PO do projeto.
- **Desenvolvimento** - Esta área está responsável pelas métricas relacionadas com o tempo de desenvolvimento da equipa. Servem de auxílio para o gestor do projeto e as métricas ajudam a perceber se é necessário realocar recursos para uma determinada área do desenvolvimento.
- ***Stakeholders*** - Esta área foca-se no produto e no desenvolvimento segundo os requisitos delineados. As métricas envolvidas nesta área avaliam se existe satisfação por parte do cliente, depois do produto ser entregue e na boa comunicação entre elementos da equipa de desenvolvimento.
- **Qualidade** - Nesta área as métricas foram selecionadas de forma a considerarem a qualidade do produto final. Analisam-se os erros praticados ao longo dos *Sprints*, procedem-se ajustes caso necessários e resolvem-se os erros ou defeitos encontrados no produto, de forma a combater uma possível insatisfação do cliente.

Na Figura 21 são apresentadas as métricas usadas em cada uma das diferentes áreas, segundo a dissertação de Luís Correia (Correia, 2018).

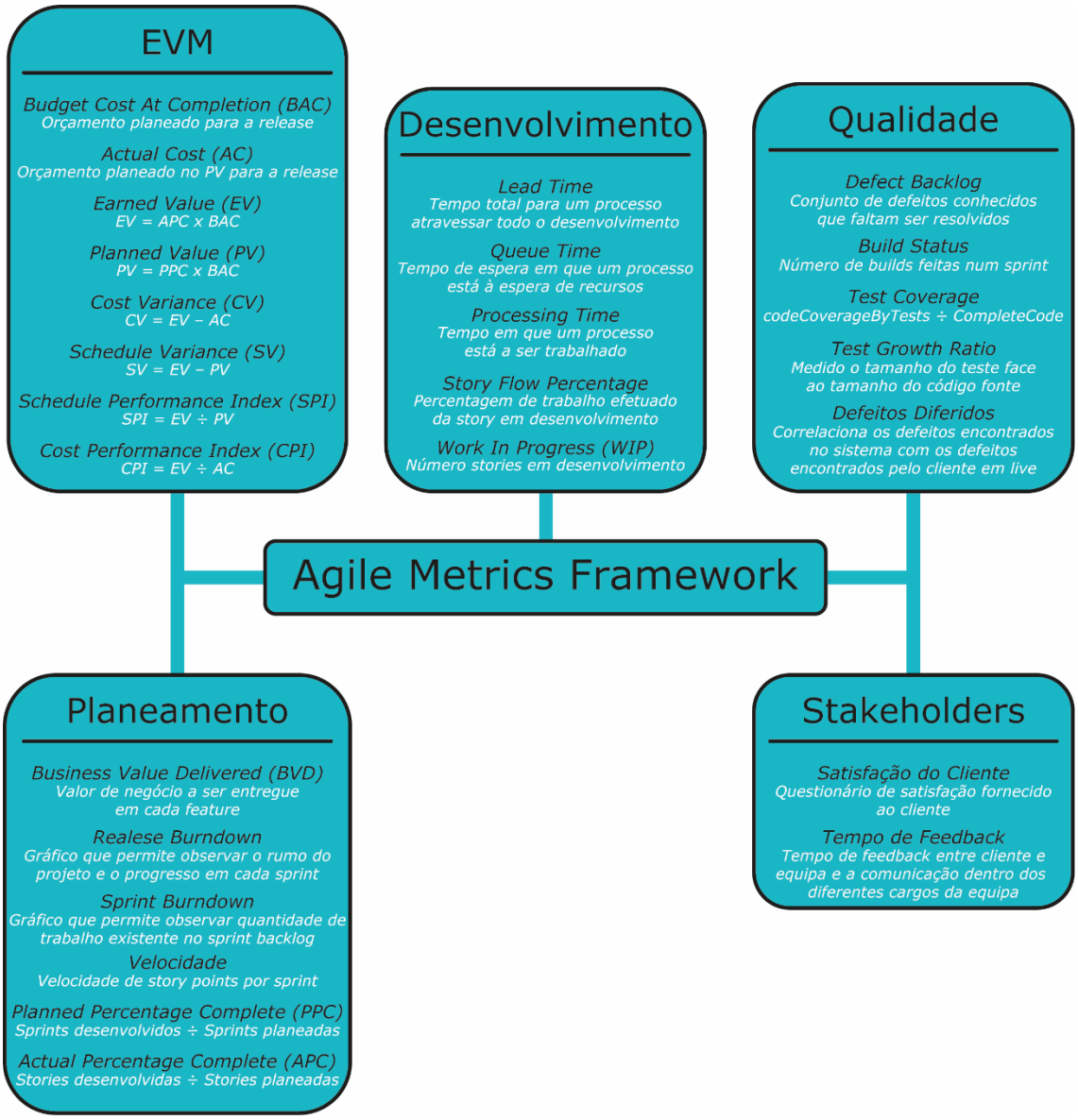

Figura 21 - Métricas Envolvidas no Agile Metrics Framework
Adaptado de: (Correia, 2018)

2.5.2 Modelo de Qualidade do Produto (ISO/IEC 25010)

A ISO 25010 ou Modelo de Qualidade do Produto pertence a uma serie de normas (ISO 250xx), desenvolvidas pela ISO (*Internacional Standards Organization),* designada de SQuaRE (Software Quality Requirements and Evaluation) (Estdale & Georgiadou, 2018).

Segundo a ISO (International Organization For Standardization, 2011), a ISO 25010:2011, divide as características de qualidade do software em dois modelos definidos da seguinte forma:

- *"A **quality in use model** composed of five characteristics (some of which are further subdivided into subcharacteristics) that relate to the outcome of interaction when a product is used in a particular context of use. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use."*
- *"A **product quality model** composed of eight characteristics (which are further subdivided into subcharacteristics) that relate to static properties of software and dynamic properties of the computer system. The model is applicable to both computer systems and software products."*

Na Tabela 4 é possível perceber quais são as características inerentes a cada um dos modelos de qualidade de software (Estdale & Georgiadou, 2018).

Tabela 4 - Características do Modelo de Qualidade ISO 25010
(Estdale & Georgiadou, 2018)

Quality in use	Product quality	Product quality (cont.)
• Effectiveness • Efficiency • Satisfaction – Usefulness – Trust – Pleasure – Comfort • Freedom from risk – Economic risk mitigation – Health and safety risk mitigation – Environmental risk mitigation • Context coverage – Context completeness – Flexibility	• Functional suitability – Functional completeness – Functional correctness – Functional appropriateness • Performance efficiency – Time behaviour – Resource utilization – Capacity • Compatibility – Co-existence – Interoperability • Usability – Appropriateness recognisability – Learnability – Operability – User error protection – User interface aesthetics – Accessibility	• Reliability – Maturity – Availability – Fault tolerance – Recoverability • Security – Confidentiality – Integrity – Non-repudiation – Accountability – Authenticity • Maintainability – Modularity – Reusability – Analysability – Modifiability – Testability • Portability – Adaptability – Installability – Replaceability

O modelo de qualidade da ISO/IEC 25010 é a base do sistema de avaliação da qualidade do produto. O modelo acima determina quais as características de qualidade que os *Stakeholders* devem ter em consideração para avaliar as propriedades estáticas do software do produto ou na interação do produto em uso. Sendo assim, a qualidade do sistema é definida por ser o grau no qual o sistema satisfaz tanto as necessidades estabelecidas pelos vários *Stakeholders* no projeto, como no valor que acrescenta ao produto. Estas necessidades dos *Stakeholders* vão de encontro às características representadas no modelo de qualidade da ISO/IEC 25010 do produto (*ISO 25010*, 2011).

2.5.3 Framework AMS

A *Framework AMS* foi desenvolvida por Douglas Neves Carlos, um aluno do curso de Ciências da Computação, da Universidade Feevale. A *Framework AMS*, cujo acrónimo advém das palavras Ágil, Métricas e *Storytelling* (AMS), tem o objetivo de apoiar o uso de métricas para acompanhar projetos de desenvolvimento, com foco em processos ágeis (Douglas Neves, 2020).

Douglas Neves (2020), desenvolveu um infográfico, como se pode observar nas Figuras 23, 24 e 25, em que ajuda a perceber como aplicar a *Framework AMS*. A *framework* está dividida em 5 fases diferentes sequenciais, o que permite a uma equipa ágil, planear, executar e melhorar, levando em conta as características dos projetos ágeis de desenvolvimento de software.

Para cada uma das fases é apresentada uma pequena descrição de como se deve proceder para aplicar a *framework*, de forma a obter resultados no acompanhamento do processo de desenvolvimento de software.

As fases definidas na *Framework AMS* são:

Figura 22 - 1ª Fase da Framework AMS
Adaptado de: (Douglas Neves, 2020)

Seleção de Métricas – Nesta etapa deve-se entender qual o contexto em que o projeto está inserido, perceber o que se quer medir e selecionar um conjunto de métricas que vão de encontro aos resultados que se pretendem obter. As métricas selecionadas pela equipa que mais se adequam ao pretendido a medir, vão acompanhar o processo de desenvolvimento.

Definição da Medição – Nesta fase, definem-se as medidas das métricas selecionadas. A equipa deve definir os procedimentos básicos para obter uma leitura proveitosa das métricas e assim permitir quantificar as medidas e melhorar com a situação. O tratamento de dados do processo pode ser realizado pelo *JIRA*, um software de gestão de projetos ágeis.

Figura 23 - 2ª e 3ª Fase da Framework AMS
Adaptado de: (Douglas Neves, 2020)

Desenvolvimento das Visualizações – Esta fase auxilia essencialmente a determinação da melhor maneira de visualizar os dados obtidos, através da aplicação das métricas selecionadas. Existem diferentes modelos visuais e pretende-se a seleção do mais adequado à situação. Knaflic (2015), evidencia no seu livro, 12 diferentes modelos gráficos com intuito de exibir a informação convenientemente, a partir das métricas selecionadas. Esses modelos visuais são Texto simples, Tabela, Mapa de calor, Gráfico de dispersão, Linha, Mapa de inclinação, Barras verticais, Barras verticais empilhadas, Cascata, Barras horizontais, Barras horizontais empilhadas e Área quadrada.

Figura 24 - 4ª e 5ª Fase de Framework AMS
Adaptado de: (Douglas Neves, 2020)

Desenvolvimento da Ferramenta – Esta fase tem como objetivo compilar os dados recolhidos, definidos nas visualizações das métricas, e tornar visíveis os resultados para todos os

elementos da equipa. É importante remover qualquer informação que não acrescente valor e assim mitigar a saturação visual.

Melhoria Contínua – Esta fase tem como objetivo recolher as visualizações das métricas selecionadas e analisá-las nos *Sprints Retrospective,* com o intuito de melhorar as áreas do projeto pretendidas. Assim, em curtos períodos de tempo, é possível identificar pontos a melhorar e traçar planos para executar essas melhorias, com o objetivo de realizar pequenas alterações ao processo que podem criar resistência à mudança.

3. CASO DE ESTUDO

3.1 Descrição

Para esta dissertação, foi realizado um estudo de caso, aplicado a uma Startup. O objetivo do estudo de caso é acompanhar, avaliar e apresentar uma proposta de melhorias, ao processo de desenvolvimento do produto, através da utilização de uma *framework* de métricas inserida na categoria de processo de desenvolvimento.

O estudo de caso foi aplicado a uma Startup, sediada em Braga, chamada *We Can Charge*. Este é um projeto que tem como foco principal, criar uma rede de carregadores de carros elétricos, descentralizada das grandes multinacionais e assim dar a capacidade a qualquer pessoa ou empresa para construir a própria rede de carregadores ou ter o seu carregador pessoal em casa.

O projeto *We Can Charge*, até à data, não está terminado e existe uma necessidade constante de apresentar novas funcionalidades, soluções e melhorias. O trabalho desenvolvido no projeto foi aplicado a alguns *Sprints* do processo de desenvolvimento. É importante mencionar que foi a primeira vez que a empresa decidiu aplicar as metodologias ágeis (em particular o Scrum), para desenvolvimento do projeto.

Para desenvolvimento do estudo de caso, foram aplicadas três fases, na *We Can Charge*:

1. **FASE I - Acompanhamento da equipa de desenvolvimento do projeto.** Nesta fase, a empresa permitiu o acesso às reuniões realizadas pela equipa, nas fases de desenvolvimento do projeto, com a condição de confidencialidade da informação partilhada e discutida, nessas reuniões. Para essas reuniões foi usada a ferramenta de videochamada, *Google Meet*, para assistir às mesmas. Esta foi a solução encontrada para resolver a impossibilidade de as reuniões serem realizadas presencialmente, devido ao estado de pandemia instalado no país pelo vírus SARS-CoV-2 (Covid-19).
2. **FASE II - Análise do processo de desenvolvimento de software, através da aplicação de uma *framework* de métricas previamente selecionada.** Nesta fase, através de um conjunto de métricas meticulosamente selecionadas, através da aplicação da Framework AMS, fez-se uma análise do processo de desenvolvimento da StartUp *We Can Charge*. Os dados obtidos foram recolhidos essencialmente, com recurso à ferramenta "JIRA

Software" e ao acompanhamento das reuniões do ciclo *Sprint*.

3. **FASE III - Apresentação de uma proposta com melhorias ao processo de desenvolvimento da *We Can Charge*.** Nesta fase, é apresentada a proposta de melhorias ao processo de desenvolvimento da Startup, com ajuda do referencial de métricas usado. O referencial pode ser aplicado a qualquer microempresa, com o objetivo de retificar, rentabilizar e melhorar o processo de desenvolvimento do projeto a trabalhar. Com a proposta de melhorias apresentado foi possível verificar os resultados inerentes a essa proposta.

3.2 Perfil da *We Can Charge*

A *We can Charge* é uma Startup fundada no final do ano de 2019, com a ideia de resolver um problema detetado na área dos carros elétricos. O projeto tem como objetivo desenvolver uma rede de carregadores de carros elétricos, descentralizada das grandes multinacionais e assim capacitar qualquer pessoa ou empresa em construir a própria rede de carregadores ou ter o seu carregador pessoal em casa. Esta ideia surgiu a partir da dificuldade existente por utilizadores de carros elétricos encontrarem postos de carregamento elétricos independentes. O projeto encontra-se em desenvolvimento e não está concluído.

A equipa integrada no desenvolvimento do projeto utiliza o Scrum como metodologia de desenvolvimento de software. Esta é a primeira vez que a Startup está a utilizar qualquer uma das metodologias de desenvolvimento para os seus projetos. Ao longo das próximas subseções são descritos os papéis, eventos e práticas relacionadas com o desenvolvimento do projeto da *We Can Charge*. Estes mecanismos são fundamentais para o bom funcionamento da metodologia Scrum, metodologia usada pela Startup.

3.2.1 Papeis

A equipa da *We Can Charge* é composta por 4 elementos, estando descritos os papéis de cada elemento e respetivas funções no projeto na Tabela 5. Os papéis dos elementos envolvidos no desenvolvimento do projeto estão em concordância com a metodologia Scrum e são *Product Owner* (PO), *Scrum Master* e *Developer Team*.

Tabela 5 - Principais Papéis na Equipa de Desenvolvimento da We Can Charge

#	Papel	Responsabilidades
1	PO	• Responsável por fazer a ponte entre a equipa e os *Stakeholders;* • Acompanha o desenvolvimento do projeto e percebe se o *Product Goal* é atingido a cada *Sprint;* • Valida o produto a cada iteração finalizada.
2	*Scrum Master* e *Developer*	• Assegura que as *guidelines* do Scrum estão a ser cumpridas; • Assegura que o projeto progride como esperado em cada *Sprint*; • Deteta e elimina entraves técnicos que possam estar a atrasar a *Development Team*; • Sempre que uma *task* passa do estado "*In Progress*" para "*Done*", faz a sua revisão; • Acumula com as funções do elemento 3.
3	*Developer*	• Seleção das suas *tasks* e contribuição na decisão do *Product Backlog* a realizar no *Sprint*; • Desenvolvimento das *tasks* do *Sprint*.
4	*Developer*	• Mesmas funções que o Elemento 3.

3.2.2 Eventos e Práticas

Durante todo o desenvolvimento do projeto, a equipa esteve em teletrabalho devido ao estado de pandemia em que o país se encontrava, resultante do surto COVID-19. Deste modo, todas as reuniões, trabalho e procedimentos foram realizados a partir da casa de cada um, com o uso de ferramentas gratuitas para realizar as comunicações e cooperar entre si.

O ***Google Meet*** foi a ferramenta de videochamada usada para realizar as reuniões dos *Sprints* do projeto. Esta é uma ferramenta de videoconferência totalmente gratuita, que fornece a possibilidade às equipas de se conectarem e poderem usufruir de funcionalidades empresariais, como se pode verificar na próxima Figura 25.

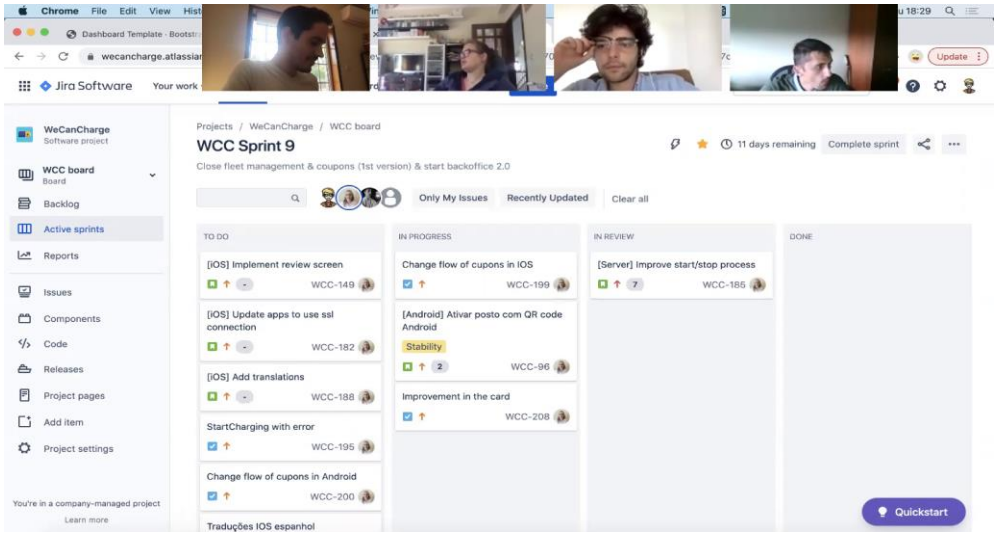

Figura 25 - Exemplo de uma Reunião Realizada a partir do Google Meet
Elaborado pelo Autor

De modo a entender da melhor forma como as reuniões foram orientadas, como o trabalho foi distribuído, como as tarefas foram executadas e como as dificuldades encontradas ao longo dos *Sprints* foram trabalhadas, foi dado acesso a essas reuniões para compreender melhor os mecanismos de funcionamento, em equipa, da *We Can Charge*.

Cada Sprint tem uma duração de 15 dias. Durante estes 15 dias são realizados três eventos da metodologia SCRUM (*Sprint Planning*, *Weekly Scrum* e *Sprint Retropective*) e adaptados para as necessidades da *We Can Charge*. Estes eventos são descritos na Tabela 6.

Tabela 6 - Sprint no Ciclo de Vida de Desenvolvimento da We Can Charge

Estrutura do *Sprint* com duração de 15 dias			
Evento	**Duração (média)**	**Periocidade**	**Características**
Sprint Planning	15 min	No início de cada *Sprint*, uma vez a cada 15 dias	• Definição do objetivo do *Sprint* a cumprir • Definição do *Sprint Backlog* e distribuição de tarefas por cada elemento • Priorizar as tarefas, segundo o nível de importância, nos três estados seguintes: Baixo, médio e alto.
Weekly Scrum	30 min	Uma vez por semana, todas as	• Expor o que foi realizado na última

		Quintas-feiras	semana de trabalho • Expor as tarefas a realizar na semana seguinte • Passar, se assim for possível, as tarefas para o estado de *Done* • Expor as dificuldades deparadas na semana
Sprint Retrospective	45 min	No final de cada *Sprint*, uma vez a cada 15 dias	• Uso da ferramenta Parabol para identificar as falhas e dificuldades ao longo do *Sprint* • Cada elemento da equipa realiza uma reflexão sobre o *Sprint*

Para auxiliar o processo de desenvolvimento do projeto e os ciclos *Sprint* é usada a ferramenta JIRA Software. Esta ferramenta desenvolvida pela empresa Atlassian permite monitorizar tarefas e realizar o acompanhamento e gestão completa, de vários projetos, em apenas um local. Com isto, se a equipa assim o quiser, o JIRA permite guardar toda a informação sobre o projeto. Desde a *Backlog List*, verificar estado do *Sprint*, transitar tarefas dentro *Sprint* ("Não iniciado" até "Concluído"), analisar relatórios referentes aos *Sprints* e às *tasks* desenvolvidas, expor problemas para o resto dos elementos da equipa do projeto, entre muitas outras funcionalidades.

Conforme referido anteriormente, cada *Sprint* tem uma duração média de 15 dias. Durante a realização desta dissertação foram acompanhados oito ciclos *Sprint* de desenvolvimento do projeto. Processo que começou no dia 19/01/2021, durante o ciclo Sprint 4 e terminou no dia 21/06/2021 com o *Sprint Retropective* do Sprint 10. Sendo assim, foi elaborada a Tabela 7, onde se pode verificar os intervalos de tempo de cada *Sprint*, onde o trabalho da dissertação recaiu.

Tabela 7 - Período de cada Sprint de Desenvolvimento do Projeto

Número do *Sprint*	**Período do *Sprint***
Sprint 4	28/12/2020 – 04/02/2021
Sprint 5	09/02/2021 – 24/02/2021
Sprint 6	24/02/2021 – 13/03/2021

Sprint 7	13/03/2021 – 27/03/2021
Sprint 8	27/03/2021 – 17/04/2021
Sprint 9	17/04/2021 – 08/05/2021
Sprint 10	08/05/2021 – 31/05/2021

Depois de apresentados os papéis, eventos e práticas envolvidas no modelo Scrum de desenvolvimento do projeto, da *We Can Charge*, estão reunidas as condições para apresentar o estudo de caso. Sendo assim, de seguida, será apresentada a análise realizada ao processo de desenvolvimento da *We Can Charge* e respetiva proposta de melhoria ao processo.

3.3 *Framework AMS* aplicada na *We Can Charge*

No seguimento do estudo realizado às várias *frameworks* de métricas aplicadas a projetos ágeis (Secção **2.5**), foi selecionada a *framework* que melhor se adapta ao trabalho pretendido a desenvolver.

Como existe o interesse de estudar e analisar o processo de desenvolvimento da *We Can Charge*, foi decidido usar como referência a *Framework AMS* (Douglas Neves, 2020), essencialmente pela sua versatilidade e forma como se adapta a diferentes projetos. Como se pôde observar na Subsecção **2.5.3**, a *Framework AMS*, é constituída por 5 fases distintas e sequenciais:

1. Seleção das Métricas;
2. Definição da Medição;
3. Desenvolvimento das Visualizações;
4. Desenvolvimento da Ferramenta;
5. Melhoria Contínua.

3.3.1 Seleção das Métricas

Nesta fase é realizada a seleção das métricas que vão ser aplicadas, no processo de desenvolvimento do projeto da *We Can Charge*. O primeiro passo é definir que características do projeto se pretendem medir. O propósito é selecionar um grupo de métricas que poderiam resolver as fragilidades que viessem a surgir no processo de desenvolvimento à medida que o trabalho fosse

progredindo. Sendo assim, o principal objetivo foi apresentar uma proposta de melhoria ao processo de desenvolvimento do projeto, com foco na análise dos seguintes pontos:

- O ritmo de trabalho da equipa;
- Progresso do projeto e capacidade de produção;
- Tarefas por completar ou iniciadas durante uma iteração;
- Cumprimento das entregas dentro do prazo.

As características acima enunciadas são referenciadas na dissertação de Raquel Pegoraro (2014), na qual se organiza e categoriza 32 métricas diferentes e se definem a sua utilidade. Este conjunto de métricas usado por Douglas Neves (2020), serve como referencial para construir a *Framework* AMS, usada também nesta dissertação. Deste modo, o conjunto de 16 métricas referenciadas na dissertação de Raquel Pegoraro (2014), na categoria de processo, especificamente selecionadas para analisar o processo de desenvolvimento são:

- *Burndown* da iteração;
- Acurácia das estimativas;
- *Burnup* do projeto;
- *Cycle Time*;
- Diagrama de Fluxo Cumulativo;
- Índice de *User Story* por iteração;
- Número de integrações por dia;
- Número de tarefas não concluídas na iteração;
- Taxa de acerto na estimativa das tarefas;
- Tempo investido em mudanças;
- Tempo investido em tarefas não planeadas;
- Tempo médio de resolução dos Impedimentos;
- *Lead Time*;
- *Throughput*;
- WIP.

Foi selecionado um conjunto restrito de métricas que vão ao encontro do objetivo e pontos em análise definidos inicialmente com a aplicação da *Framework AMS*. Para continuar a aplicar a *framework*, foram selecionadas (do conjunto de métricas acima) segundo o referencial desenvolvido por Pegoraro (2014), as seguintes métricas, exibidas na Tabela 8.

No conjunto de métricas selecionadas, decidiu-se integrar a métrica *Velocity*, da categoria de Equipa (Pegoraro, 2014), devido à sua importância na visualização da gestão do trabalho (comprometido a realizar e completo) ao longo dos *Sprints*, como forma de verificar o fluxo de trabalho em cada *Sprint* e obter uma visão geral do estudo realizado.

Tabela 8 - Conjunto de métricas aplicadas à We Can Charge

Métrica	Objetivo da métrica
***Burndown* da iteração**	Verificar o progresso das tarefas realizadas pela equipa no *Sprint*.
***Burnup* da Iteração**	Quantificar o trabalho restante para atingir a meta do *Sprint*.
Diagrama Fluxo Cumulativo	Verificar o estado do projeto.
Velocity	Verificar a quantidade de trabalho que a equipa de desenvolvimento consegue entregar num *Sprint*.
Cycle Time	Verificar o tempo de ciclo desde que uma tarefa começa a ser trabalhada até estar completamente finalizada.
Thoughput	Verificar o número médio de tarefas entregues por *Sprint*.
WIP	Monitorização do número médio de tarefas em progresso por *Sprint*.

Sendo assim, segundo Pegoraro (2014), o conjunto de métricas selecionado é constituído por 6 métricas da categoria de processo e 1 métrica da categoria de Equipa. Conforme referido anteriormente, este conjunto restrito de métricas ajudará a criar uma proposta de melhoria no processo de desenvolvimento do projeto.

3.3.2 Definição da Medição

Nesta fase da *Framework AMS*, é definido o modo como as métricas são medidas, com base no trabalho desenvolvido por Pegoraro (2014), como se pode observar na Tabela 9.

Tabela 9 - Definição das medidas das métricas

Métrica	Unidades	Fonte de Recolha	Apresentação
***Burndown* da iteração**	*Story Points*	JIRA Software	Gráfico
***Burnup* da Iteração**	*Story Points*	JIRA Software	Gráfico
Diagrama Fluxo Cumulativo	Número de Pendências	JIRA Software	Gráfico
Velocity	*Story Points*	JIRA Software	Gráfico
Cycle Time	Tempo decorrido	JIRA Software	Gráfico
Thoughput	Número de Pendências	JIRA Software	Gráfico
WIP	Número de Pendências	JIRA Software	Gráfico

Para obter dados relativos às métricas selecionadas foi usado como referência, o JIRA Software, um software especializado em gestão e controlo de projetos, como explicado anteriormente.

3.3.3 Desenvolvimento das Visualizações

Com as métricas e medições definidas nas etapas anteriores, é o momento de definir como estas serão visualizadas. De seguida, será definida para cada métrica um modelo gráfico, facilitando visualmente na leitura e extração de informação das métricas selecionadas. Na Tabela 10 poderá observar-se os modelos usados para cada métrica, segundo Knaflic (2015).

Tabela 10 - Definição das características gráficas das métricas

Métrica	Modelo Gráfico	Eixo x	Eixo y
***Burndown* da iteração**	Modelo de linhas	Tempo (Dias do *Sprint*)	*Story Points*
***Burnup* da**	Modelo de linhas	Tempo (Dias do	*Story Points*

Iteração		*Sprint*	
Diagrama Fluxo Cumulativo	Modelo de Área Quadrada	Tempo (Período selecionado)	Número de Pendências
Velocity	Modelo de Barras Verticais	Número do *Sprint*	*Story Points*
Cycle Time	Modelo Gráfico de dispersão	Data de Transição do Item	Tempo Percorrido (Dias)
Thoughput	Modelo de Área Quadrada	Tempo (Período selecionado)	Número de Pendências
WIP	Modelo de Área Quadrada	Tempo (Período selecionado)	Número de Pendências

Na Figura 26 é possível observar exemplos dos modelos gráficos usados neste conjunto de métricas.

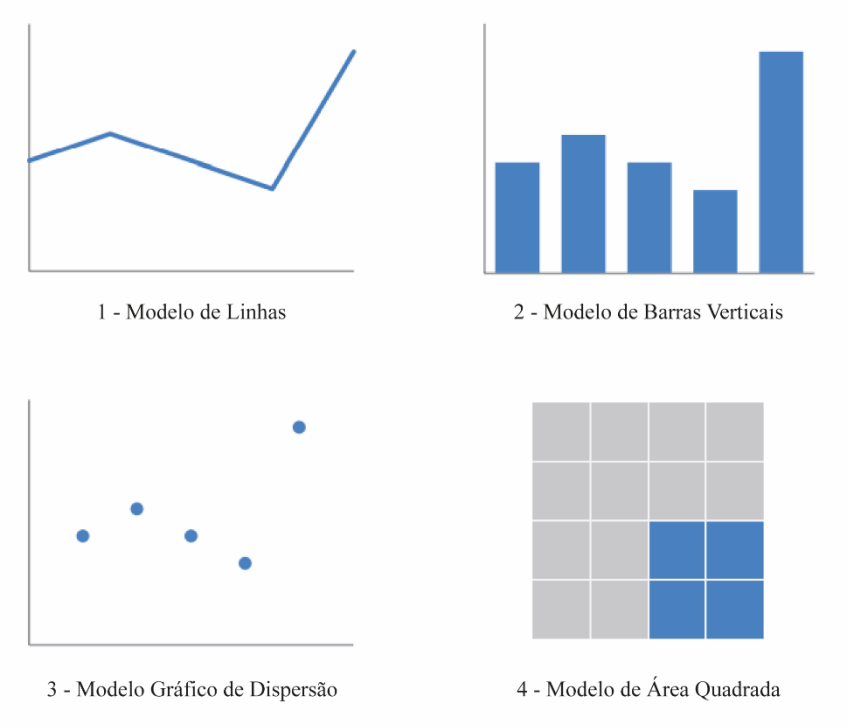

Figura 26 - Modelos gráficos usados no conjunto de métricas,
Adaptado de: (Knaflic, 2015)

3.4 Apresentação dos Resultados

Para efeitos do que foi proposto a realizar nesta dissertação e estudo de caso, nesta etapa não se desenvolveu um aplicativo para gerar as visualizações. Como a *We Can Charge* já utiliza o JIRA Software para efeitos de gestão do projeto, o software gera automaticamente gráficos e guarda toda a informação relacionada com os *Sprints* do projeto. Sendo assim, não existe a necessidade de criar uma nova ferramenta/aplicativo, quando se pode utilizar as ferramentas já usadas pela *We Can Charge*.

Conforme referido anteriormente, nesta dissertação realizou-se o acompanhamento de oito *Sprints*, com o objetivo de analisar e melhorar o processo de desenvolvimento do projeto. Sendo assim, depois de aplicar a *Framework AMS*, definir as métricas, definir a medição e definir o desenvolvimento das visualizações, estão reunidas as condições para apresentar a análise realizada e consequentemente os resultados extraídos.

Os *Sprints* foram divididos em dois grupos. Do Sprint 4 ao Sprint 7 e do Sprint 8 ao Sprint 10. Os *Sprints* foram separados em dois grupos pois foi aplicada a proposta de melhorias ao processo entre o Sprint 7 e o Sprint 8, no seguimento da análise e aplicação das métricas selecionadas ao primeiro grupo de *Sprints*. Depois de apresentada a proposta com as melhorias ao processo de desenvolvimento são verificados os resultados obtidos da implementação da proposta no grupo II de *Sprints*, entre o Sprint 8 e o Sprint 10. Sendo assim, estão reunidas as condições para analisar as visualizações geradas.

3.4.1 Grupo I

Conforme referido anteriormente, o estudo realizado a partir das métricas selecionadas foi dividido em dois grupos. A análise do processo de desenvolvimento de cada um dos *Sprints* é realizado com o auxílio de cada uma das métricas em estudo.

O grupo I é constituído pelos primeiros 4 *Sprints* em estudo. Todos os gráficos foram retirados do JIRA Software, onde é armazenada a informação em relação ao projeto, *We Can Charge*.

Burndown da Iteração

A métrica *Burndown* da Iteração usa o modelo de linhas, em que o eixo x, representa o tempo (ciclo *Sprint* em dias) e o eixo y, os *Story Points* do *Sprint*. A Figura 27, está dividida em quatro momentos, em que cada um representa cada um dos *Sprints* em estudo. O mesmo acontece em algumas das seguintes métricas em estudo.

Através da análise dos gráficos da Figura 27 é possível perceber que existem duas linhas com cores diferentes. Uma, a cinzento, que serve com guia/diretriz do *Sprint*, considerando o número de *Story Points* e dias para finalizar o *Sprint* (idealmente). Uma, a vermelho, que representa o número de *Story Points* completos durante o *Sprint*.

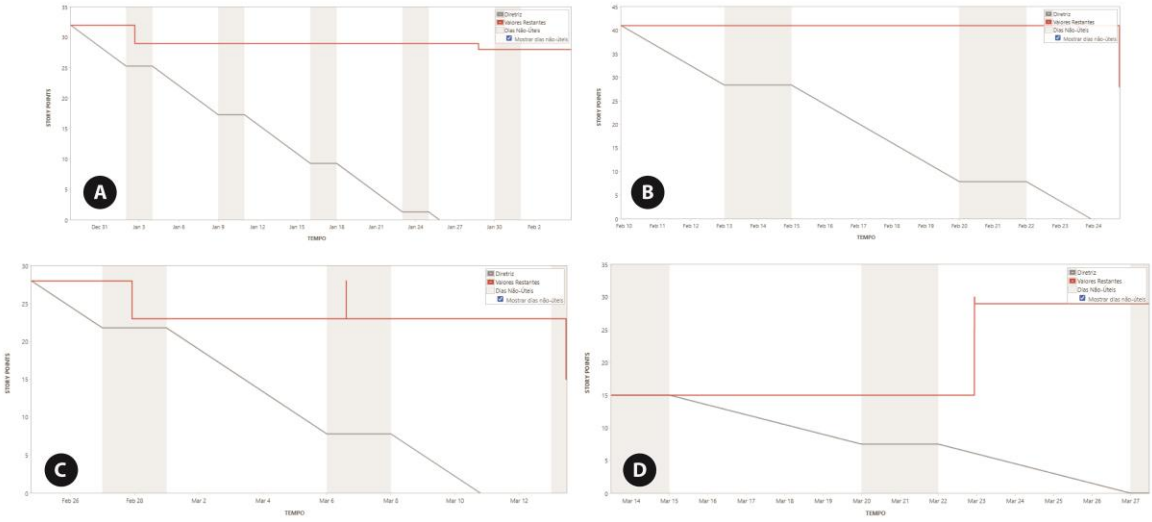

Figura 27 - Gráficos Burndown da Iteração: (A) - Sprint 4; (B) - Sprint 5; (C) - Sprint 6; (D) - Sprint 7
Elaborado pelo Autor

Idealmente a linha vermelha deveria acompanhar o comportamento da linha cinzenta e tender para zero ao longo do tempo, o que nem sempre aconteceu.

Esta é uma métrica bastante útil para observar se as tarefas estão a ser entregues dentro dos prazos estabelecidos pela equipa (Pegoraro, 2014). Como se pode perceber, pela leitura de cada um dos gráficos, o comportamento não foi ideal em cada um dos *Sprints*, nesta primeira fase. O que acaba por ser normal, segundo as características em que o projeto é realizado:

- Primeiro projeto que a empresa utiliza o Scrum como ferramenta de apoio ao projeto;
- Equipa pequena (3 elementos), com múltiplos trabalhos;
- Definição de novos objetivos e novas diretrizes a cada *Sprint* (mudança do rumo do projeto);
- Desleixo em estimar o esforço em *Story Points* para definir trabalho nas tarefas (apesar de ainda se concluir um número significativo de tarefas, principalmente no Sprint 4 e 5);
- Tentativa em dominar a ferramenta JIRA Software. Como se pode perceber na leitura dos gráficos, os maiores *stepdowns* são efetuados no final dos *Sprints*, na tentativa de ajustar as tarefas já concluídas;
- O projeto sofre paragens esporádicas, de curtos períodos de tempo (entre 1 a 2 semanas), muitas vezes derivado da espera de respostas por parte dos clientes, que levariam a caminhos

diferentes no rumo do projeto.

Burnup da Iteração

Esta métrica é igualmente representada graficamente pelo modelo de linhas, em que o eixo x, é medido em dias e o eixo y em *Story Points*. Cada gráfico representa um *Sprint* diferente, estruturalmente bastante similar à métrica anterior em estudo. Neste caso, a métrica em estudo é representada por três linhas, com três cores distintas. A linha vermelha define o alcance geral do *Sprint*, ou seja, o número total de *Story Points* no *Sprint*. A linha cinzenta define a diretriz/guia no *Sprint*, o ritmo ideal em termos de conclusão das tarefas propostas para o *Sprint*. Por outro lado, a linha verde define o número real de *Story Points* concluídos no *Sprint*. Como na métrica anterior, idealmente, a linha verde deveria acompanhar a evolução da linha cinzenta e intersetar-se com a linha vermelha no final, o que significaria que ficou completo o trabalho proposto para o *Sprint*. Apesar da perspetiva revertida, o objetivo desta métrica é bastante similar à anterior, com foco em prever o cumprimento das entregas dentro dos prazos estabelecidos (Pegoraro, 2014).

Como se pode observar a partir da leitura dos gráficos na Figura 28, em nenhum dos *Sprints* o número de tarefas completas atingiu o alcance do *Sprint*. Além disso, a evolução e ritmo das *Story Points* completas desenrolaram-se de forma um pouco lenta.

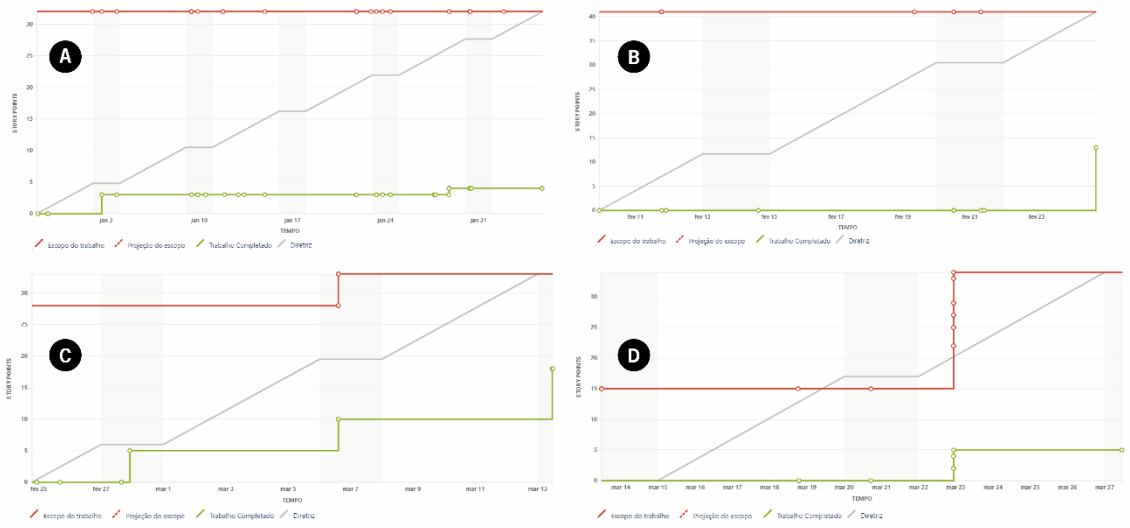

Figura 28 – Gráficos Burnup da Iteração: (A) - Sprint 4; (B) - Sprint 5; (C) - Sprint 6; (D) - Sprint 7
Elaborado pelo Autor

Diagrama de Fluxo Cumulativo (DFC)

A visualização do DFC é representada na forma de um modelo de Área Quadrada. No diagrama, da Figura 29, da métrica em estudo, o eixo x representa o tempo (em dias), e o eixo y, o

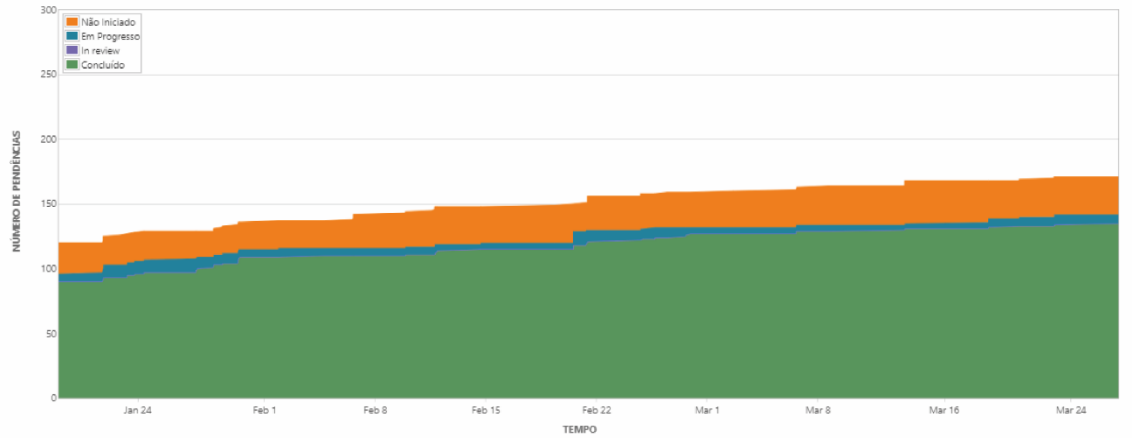

Figura 29 - Diagrama Fluxo Cumulativo correspondente ao grupo I de Sprints
Elaborado pelo Autor

número de tarefas do projeto. Como se pode observar pela análise da figura, o diagrama contém áreas com diferentes cores que definem diferentes estados do projeto. Os estados são representados de cima para baixo, da cor laranja para a verde, do estado "não iniciado" para "concluído", passando pelo estado "em progresso".

A métrica tem como objetivo visualizar o trabalho do projeto de uma forma geral, conhecer o ritmo da equipa e perceber o estado das tarefas no projeto (Pegoraro, 2014). Para além disso é possível desconstruir o DFC, de forma, a retirar métricas como o WIP e o *Thoughput*, métricas essas também em estudo. Este é um processo que se pode realizar automaticamente, a partir do manuseamento dos parâmetros do JIRA Software e retirar desta forma as representações gráficas apresentadas nas Figura 32 e Figura 34.

Cycle Time

O *Cycle Time* é uma métrica, como se pode observar pela Figura 30, usa o modelo de dispersão em que o eixo x, representa a data de transição do estado dos itens e o eixo y, o tempo decorrido em dias para essa transição acontecer.

A métrica mede o tempo desde que um item é iniciado até ao momento em que fica concluído e é representado pelos círculos verdes, na forma de ocorrências. A linha vermelha mostra a média geral do *Cycle Time* das ocorrências, a linha azul a média móvel entre ocorrências e a área azul, o desvio padrão entre as ocorrências.

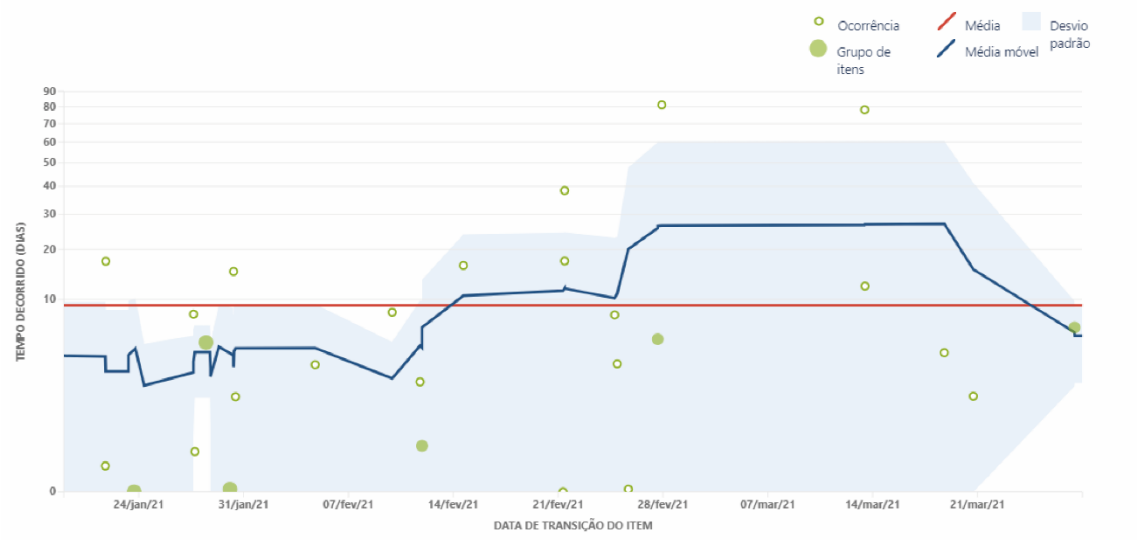

Figura 30 -Gráfico de Dispersão do Cycle Time correspondente ao grupo I de Sprints
Elaborado pelo Autor

Com isto, é importante reter através da análise da Figura 30, as evidências seguintes:

- A média de ocorrências é aproximadamente 10 dias;
- O desvio padrão no gráfico deveria reduzir e não dispersar ao longo do projeto, com os ajustes que poderiam ser feitos entre *Sprints*;
- Com a melhoria do processo, a média móvel deveria diminuir para aumentar a taxa de resultados e eficiência de desenvolvimento;
- Poucas ocorrências com valores atípicos o que é um sinal positivo.

Thoughput

O *Thoughput* é uma métrica extraída a partir do estudo do DFC. Depois de selecionado o período pretendido no JIRA Software, correspondente a cada *Sprint*, desseleciona-se da leitura do gráfico de DFC os estados, "Não iniciado" e "Em progresso", deixando o número de tarefas concluídas, do estado "Concluído". Como se pode observar na Figura 31.

Figura 31 - Exemplo da manipulação do gráfico DFC para obter o Thoughput
Elaborado pelo Autor

Como se pode observar pela Figura 32, esta métrica usa o Modelo de Área Quadrada em que o eixo x, representa tempo (dias) e o eixo y, o número de pendências concluídas.

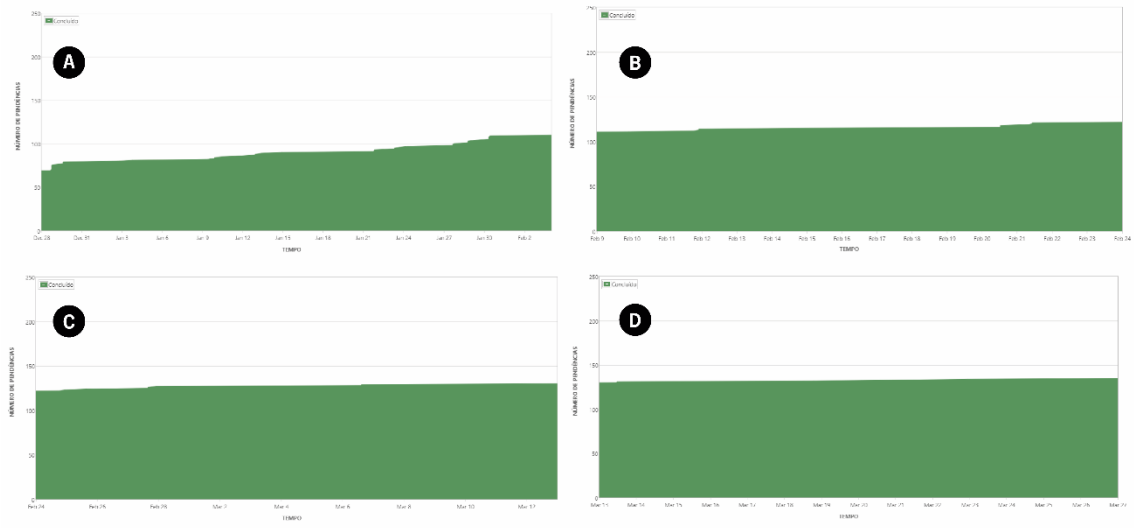

Figura 32 - Gráficos Thoughput: (A) - Sprint 4; (B) - Sprint 5; (C) - Sprint 6; (D) - Sprint 7
Elaborado pelo Autor

Para cada um dos quatro *Sprints* deste Grupo I, é apresentado na Tabela 11, o número de pendências concluídas.

Tabela 11 - Nº de Pendências concluídas por Sprint, no Grupo I

Sprint	**Sprint 4**	**Sprint 5**	**Sprint 6**	**Sprint 7**
Nº pendências concluídas	34	12	9	4

O *Thoughput* deveria aumentar ao longo dos *Sprints* e não diminuir ou até mesmo estabilizar, mas devido algumas das razões apresentadas na métrica *Burndown* da iteração é justificável os valores apresentados.

WIP

O WIP é também uma métrica extraída a partir do tratamento do DFC. Neste caso, os estados "Não Iniciado" e "Concluído" não são apresentados e apenas é deixado visível na visualização do JIRA software, o estado "Em Progresso", retirando daí os dados necessários que definem o WIP. Como é apresentado no exemplo da Figura 34.

Figura 32 - Gráficos WIP: (A) - Sprint 4; (B) - Sprint 5; (C) - Sprint 6; (D) - Sprint 7
Elaborado pelo Autor

Nesta métrica é usado o modelo de Área Quadrada, como se pode observar na Figura 35, em que o eixo x é representado em tempo (dias) e o eixo y, em número de pendências (em progresso).

O gráfico sofre flutuação significativa se estiverem a entrar e sair tarefas do estado "em progresso", pelo contrário, a área do gráfico mantem-se contínua se não forem adicionadas tarefas ao *Sprint* ou não existirem saída de tarefas "em progresso" para "concluído".

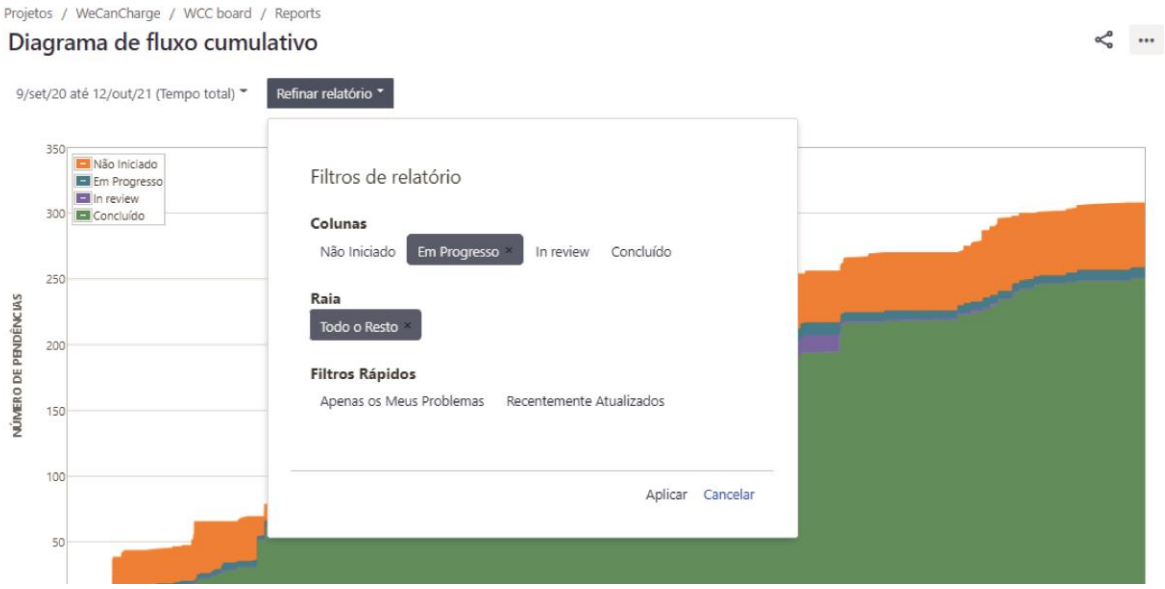

Figura 33 - Exemplo da manipulação do gráfico DFC para obter o WIP
Elaborado pelo Autor

Para esta métrica foi determinado o número médio de pendências em progresso em cada um dos *Sprint* (com o auxílio de uma folha de cálculo no Excel), de forma a perceber se existe uma variação significativa entre *Sprints*. O resultado pode ser observado na Tabela 12.

Tabela 12 - Nº médio de pendências em progresso por Sprint

Sprint	**Sprint 4**	**Sprint 5**	**Sprint 6**	**Sprint 7**
Nº médio de pendências em progresso	7.22	7.08	7.25	6.5

Na leitura da tabela é possível perceber que não existe uma variação significativa, no número médio de pendência em progresso, entre *Sprints*.

3.4.2 Proposta de melhoria do processo de desenvolvimento

No seguimento do acompanhamento e análise dos primeiros quatro *Sprints* em estudo, considerou-se que era necessário apresentar uma proposta de melhoria do processo de desenvolvimento do projeto da *We Can Charge*, visto que foram verificadas algumas falhas. As falhas aconteceram naturalmente, essencialmente, pela falta de experiência da equipa em usar o Scrum como metodologia de apoio.

A proposta foi apresentada no *Sprint Retrospective* do Sprint 7 e foi construída em cooperação com a equipa de desenvolvimento do projeto em estudo. Todos os pontos mencionados na proposta de melhoria surgiram como forma de otimizar e organizar o processo de desenvolvimento e rentabilizar o tempo de cada elemento na equipa. Sendo assim, as falhas observadas e enunciadas nos *Sprints* em estudo do grupo I, foram as seguintes:

1. As tarefas não estão a ser corretamente criadas:
    - As tarefas não são estimadas;
    - Não se sabe o esforço real para cada tarefa ou a capacidade que a equipa necessita de despender;
    - Tarefas não têm uma descrição sucinta e esclarecedora sobre o tema que pretendem resolver;
    - Algumas das tarefas estão carentes de responsável;
    - Organizar as tarefas por *labels*, relacionando a área do projeto onde se está a trabalhar com o problema que será resolvido (ex: [IOS] – problemas relacionados com IOS, [SERVER] – problemas com o servidor, [Hardware] – problemas com o hardware e etc.).
2. As tarefas são demasiado grandes:
    - As tarefas estão a passar de um *Sprint* para o outro consecutivamente;
    - A equipa perde a visibilidade em relação ao trabalho que é necessário fazer ou foi feito de uma tarefa.
3. Falta de visibilidade na globalidade do processo e são apenas 4 elementos na equipa:
    - Não é possível perceber o que cada um anda a fazer no projeto assincronamente;
    - Através da observação direta do quadro no JIRA Software deveria ser possível saber o que cada um está a fazer (em termos de trabalho) e nem sempre é possível.
4. As tarefas do gestor de projetos no quadro de tarefas trazem mais problemas do que ajuda:
    - Precisam de muito tempo para as estimar corretamente;
    - Algumas das tarefas não conseguem ser realizadas dentro dos limites de tempo razoáveis;

- As tarefas estão a afetar o comportamento do *Sprint*.

5. Definição de *Done:*

    - Não há definição de *Done.* Passar a defini-lo em cada tarefa;
    - Cria discórdia entre elementos da equipa, em relação à integração de módulos no projeto.

Através da análise realizada, nos *Sprints* anteriores, foi possível apresentar estes cinco pontos, que debilitam o processo de desenvolvimento do projeto. Sendo assim, para tentar colmatar as falhas anteriores, foram apresentadas as seguintes sugestões de melhoria:

1. Definir as tarefas é responsabilidade de cada um dos membros da equipa:

    - Assegurar um prazo final (de duas em duas semanas) em que todas as tarefas devem ser preenchidas para essa semana;
    - As tarefas não devem durar mais de uma semana no quadro, se for o caso, a tarefa deve ser dividida em sub-tarefas mais pequenas.
    - Todas as tarefas devem estar o mais completas possível, com toda a informação necessária e estimá-las corretamente.

2. Definir o *Done*:

    - Quando a tarefa estiver realmente finalizada, defini-la como *Done*;
    - A tarefa deve ser inicializada com o preenchimento do cartão de tarefa e terminada quando realmente for finalizada;
    - Incluir a o estado "*In Review*" no quadro de tarefas, em que tudo é testado, antes de passar a *Done,* e assim ter a certeza de que as tarefas foram finalizadas com sucesso.

3. O Gestor de projetos define as tarefas no *Backlog:*

    - O gestor tem a visibilidade e por isso deve criar e acrescentar as tarefas se assim for necessário;
    - Independentemente de quem fica com uma tarefa incompleta, devem ser preenchidos todos os detalhes da tarefa antes de a iniciar.

4. O progresso realizado por parte do gestor de projetos deve ser seguido num documento próprio:

- Com esta medida torna-se possível perceber quais os acordos que estão em cima da mesa, como estão a ser geridos, quais são os prazos e o que esperar de cada acordo. Sendo assim desimpede o quadro de tarefas com tarefas que não têm muitas vezes um prazo de validade;
- As tarefas do gestor de projetos não fazem parte do *Backlog;*
- Todas as comunicações, reuniões e outras tarefas podem ser registadas na forma de um diário para manter a par a equipa sobre como os acordos estão a ser conduzidos. Ter acesso a esta informação é mais relevante que perder o tempo a criar *taks cards* no *Backlog*.

Com as melhorias apresentadas após análise do grupo I de *Sprints*, foi possível verificar os resultados positivos obtidos no processo de desenvolvimento do projeto no grupo II de *Sprints*. Estas melhorias podem ser observadas nas seguintes métricas aplicadas ao grupo II.

3.4.3 Grupo II

Neste segundo grupo foi realizado o estudo das mesmas métricas estudadas no primeiro grupo, mas para os restantes três *Sprints* em estudo. Todos os gráficos e dados em estudo foram retirados do JIRA Software e do acompanhamento da equipa ao longo dos cinco meses.

Burndown da Iteração

Na análise dos seguintes gráficos da Figura 36, do *Burndown* da iteração, pode-se retirar que parece existir uma ligeira melhoria no processo de desenvolvimento, em relação aos *Sprints* em estudo no grupo I. Pode-se observar que:

- Um esforço maior em catalogar e distribuir *Story Points* pelas tarefas em execução, nos diferentes *Sprints*;
- A linha vermelha correspondente aos valores restantes do *Sprint* (gráfico A e B) tende a decrescer e a cumprir o objetivo do *Sprint* em concluir todas as tarefas. Por outro lado, no gráfico C, esta tendência não foi de todo cumprida. Isto acontece devido aos ajustes criados nos objetivos e *Story Points* das tarefas do *Sprint;*

- Fragmentação das tarefas em sub-tarefas mais pequenas foi uma forma de resolver o problema da estagnação dentro dos *Sprints*, como se observava no grupo I.

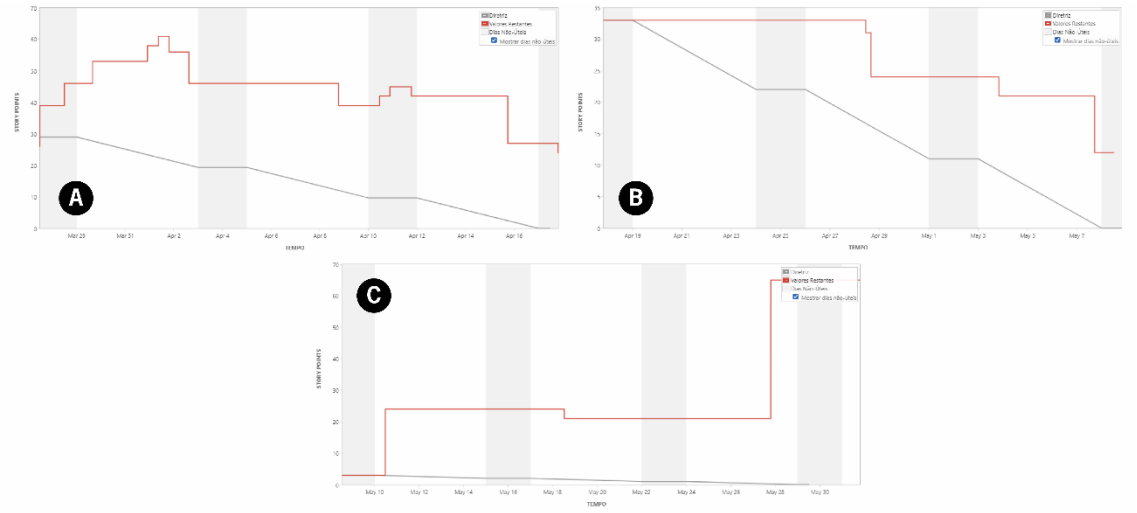

Figura 34 - Gráficos Burndown da Iteração: (A) - Sprint 8; (B) - Sprint 9; (C) - Sprint 10
Elaborado pelo Autor

Burnup da Iteração

Nos gráficos *Burnup* da Iteração, apresentados abaixo na Figura 37, pode-se também observar melhorias em relação aos gráficos da Figura 28, do grupo I.

Essencialmente denota-se o uso de *Story Points* nas tarefas definidas. O número *Story Points* completas subiu consideravelmente, aproximando-se das *Story Points* definidas no projeto, principalmente no gráfico A e B.

Quanto ao gráfico C, pode-se analisar o ajuste dos *Story Points* realizado no final de *Sprint*, como se referiu na métrica anterior. O *sprint* estagnou e não acompanhou o bom desempenho dos anteriores *Sprints*.

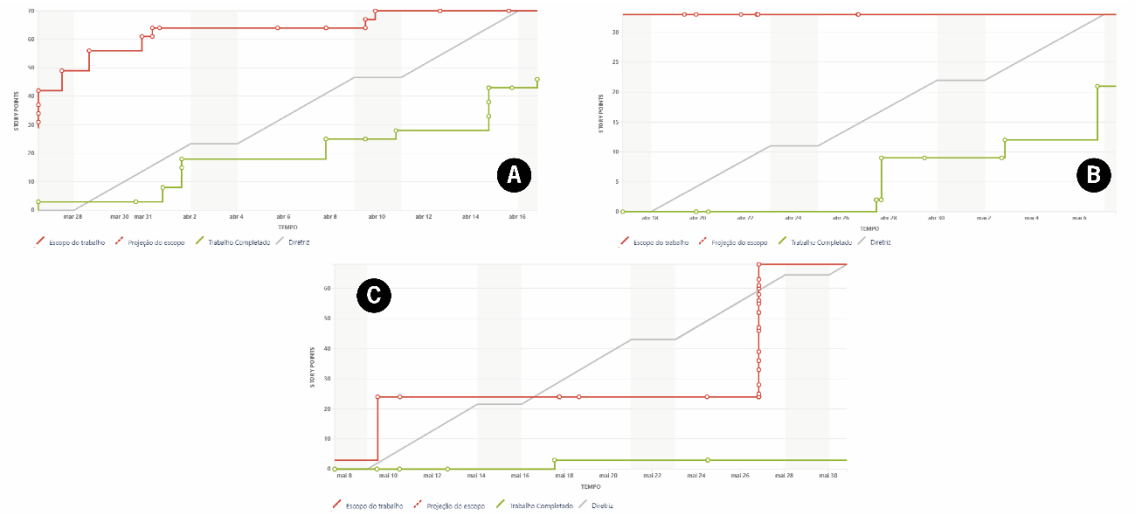

Figura 35 - Gráficos Burnup da Iteração: (A) - Sprint 8; (B) - Sprint 9; (C) - Sprint 10
Elaborado pelo Autor

Diagrama Fluxo Cumulativo

No DFC é possível ter uma visão geral do processo de desenvolvimento, como se pode perceber pela Figura 38.

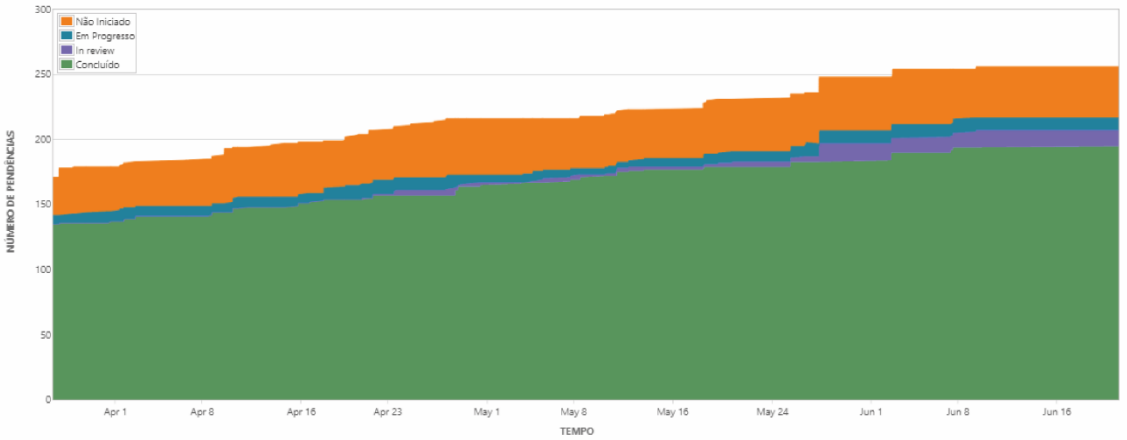

Figura 36 - Diagrama Fluxo Cumulativo correspondente ao grupo II de Sprints
Elaborado pelo Autor

Uma das características que visivelmente se destaca é a maior atividade geral no processo em relação ao gráfico DFC, do grupo I. Para além disso, é possível observar que foi adicionado um quarto estado ao processo, designado "*In Review*". Este estado foi um acrescento útil ao processo, já que um elemento da equipa fica responsável por rever todo o trabalho que transita do estado "Em Progresso" para o estado "completo", o que evita erros e falhas na entrega e conceção do produto final.

Cycle Time

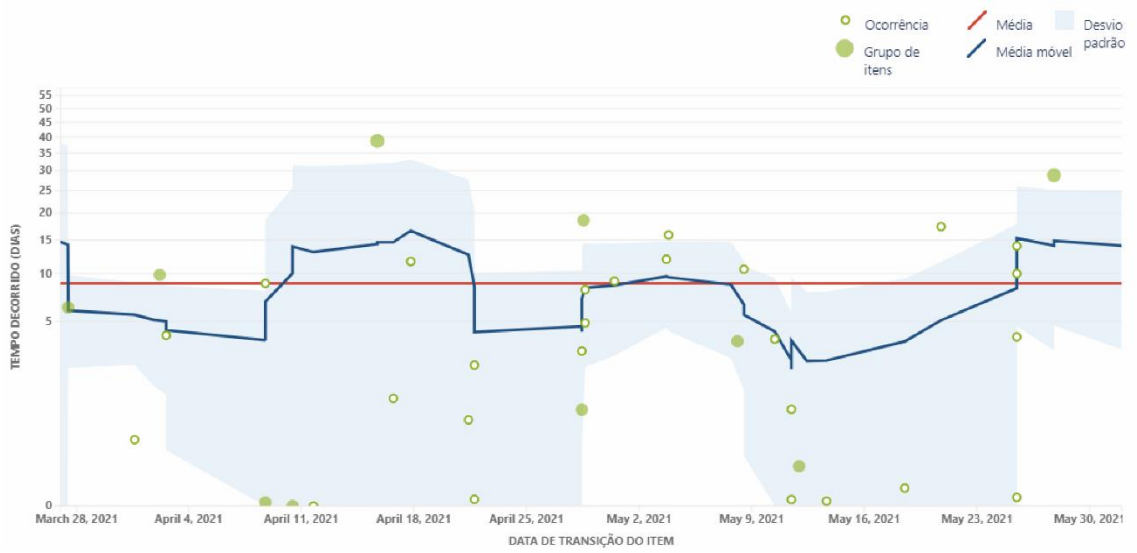

Figura 37 - Gráfico de Dispersão do Cycle Time correspondente ao grupo II de Sprints
Elaborado pelo Autor

Este é mais um exemplo da existência de uma significativa melhoria em relação ao *Cycle Time* do grupo I. Algumas das observações apontadas no grupo I foram ligeiramente corrigidas com as melhorias apresentadas ao processo. Através da leitura da Figura 39 é possível perceber que:

- O número de ocorrências com valores atípicos diminuiu, mantendo-se abaixo ou bastante próximas da linha média de ocorrências;
- O desvio padrão concentra-se em áreas mais especificas, não flutuando grosseiramente, o que evidência melhorias no processo e controlo do *Cycle Time* das tarefas e na sua previsibilidade;
- Pelo contrário, a média móvel, mostra que não foram feitos os possíveis para apresentar uma melhor taxa de resultados, já que não existem melhorias significativas em relação ao grupo I.

Thoughput

No *Thoughput*, como acontece no grupo I, os gráficos representam o número de pendências completas ao longo dos últimos três *Sprints* em estudo. Para isso, é criada a Tabela 13, para precisar o número de tarefas completas através do uso e análise do JIRA Software.

Tabela 13 - Nº de Pendências concluídas por Sprint, no Grupo II

Sprint	**Sprint 8**	**Sprint 9**	**Sprint 10**
Nº pendências concluídas	16	13	10

Na tabela anterior, é possível observar que existe uma melhoria em relação à Tabela 11, correspondente ao grupo I analisado. Apesar do número de pendências resolvidas diminuir ao longo do *Sprint*, mostra que também não é um decréscimo significativo. Existe uma maior consistência comparativamente com o grupo I.

O número de pendências resolvidas ao longo dos *Sprints*, também pode estar conectada diretamente com o uso regular das *Story Points* e a fragmentação das tarefas em sub-tarefas.

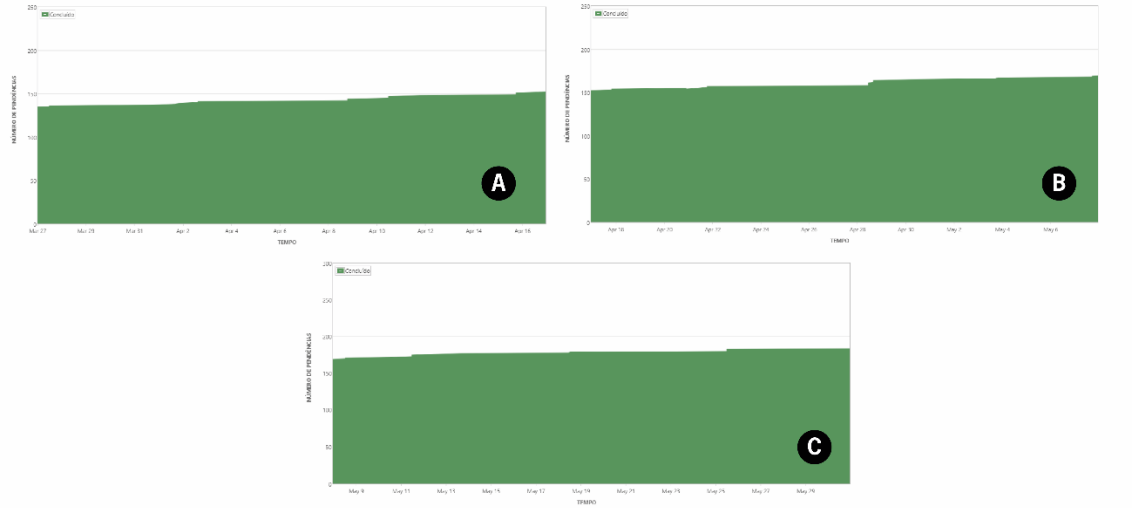

Figura 38 - Gráficos Thoughput: (A) - Sprint 8; (B) - Sprint 9; (C) - Sprint 10
Elaborado pelo Autor

WIP

Para esta métrica espera-se perceber, em relação ao grupo I, se houve um aumento ou diminuição do número médio de pendências em progresso.

Tabela 14 - Nº Médio de Pendências em Progresso por Sprint no Grupo II

Sprint	**Sprint 8**	**Sprint 9**	**Sprint 10**
Nº médio de pendências em Progresso	7.87	10.57	11.17

Com a leitura da Tabela 14 é possível perceber, que em comparação com o grupo I, o número médio de pendências em progresso aumentou. Isto acontece devido ao ajuste realizado ao processo de desenvolvimento, acrescentando o estado "*In Review*", gerando um maior fluxo de tarefas em transição para o estado de completas.

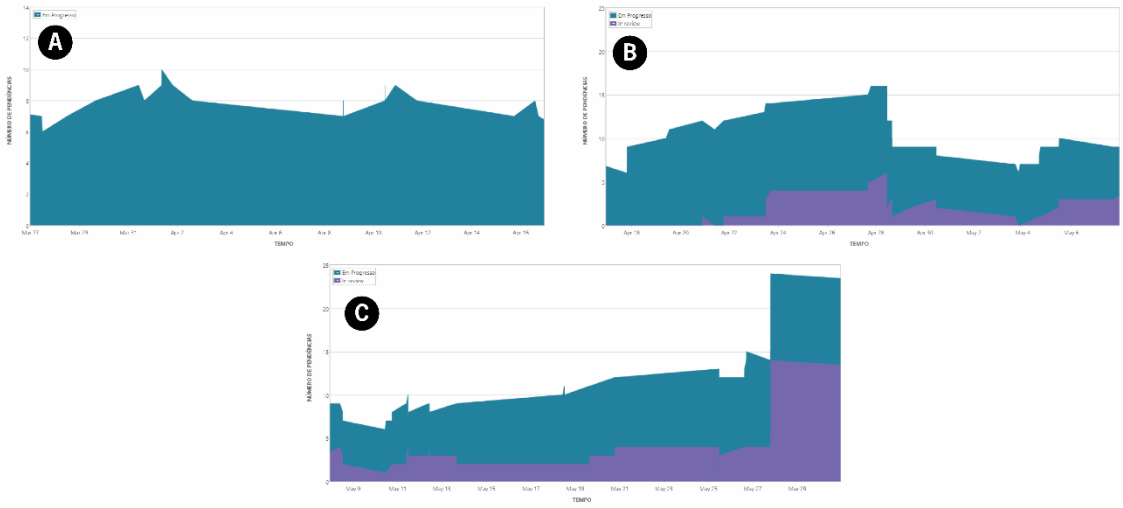

Figura 39 - Gráficos WIP retirados do JIRA Software: (A) - Sprint 8; (B) - Sprint 9; (C) - Sprint 10
Elaborado pelo Autor

3.4.4 Visão Geral do Projeto

Na Figura 42, é possível obter uma visão geral dos *Sprints* estudados, durante a realização desta dissertação.

A *velocity* é uma métrica em que é aplicada a visualização do Modelo de Barras Verticais, extraído como todos os outros dados e gráficos do estudo realizado, do JIRA Software. No gráfico da métrica *velocity,* o eixo x é dimensionado em *Sprints* e o eixo y representa as *Story Points* do projeto. Em relação às barras existem duas cores distintas, uma barra de cor cinzenta e a outra de cor verde, uma representa o número de *Story Points* comprometidos a realizar nos *Sprints* (barra cinzenta) e a outra, os *Story Points* completos nos *Sprints* (barra verde).

A *velocity* é uma métrica que representa a quantidade de trabalho que a equipa de desenvolvimento conseguiu entregar numa iteração, dimensionando consecutivamente os seguintes *Sprints.* Esta acaba por ser uma métrica de apoio bastante útil para o controlo do projeto e dos seguintes *Sprints* em execução (Pegoraro, 2014).

Observando o gráfico da Figura 42, é possível perceber que existe um controlo e maior rigor no cumprimento dos compromissos assumidos para os *Sprints* do grupo II (Sprint 8,9 e 10), do que no grupo I (Sprint 4,5,6 e 7). O fato de se ter aplicado e apresentado melhorias ao processo entre os dois grupos, traduziu-se em benefícios para desenvolvimento de cada um dos *Sprints*. A experiência adquirida em cada um dos *Sprints,* por parte da equipa, levou a que houvesse um aprimoramento dos *Sprints* seguintes, resultando em melhores resultados e melhor funcionamento.

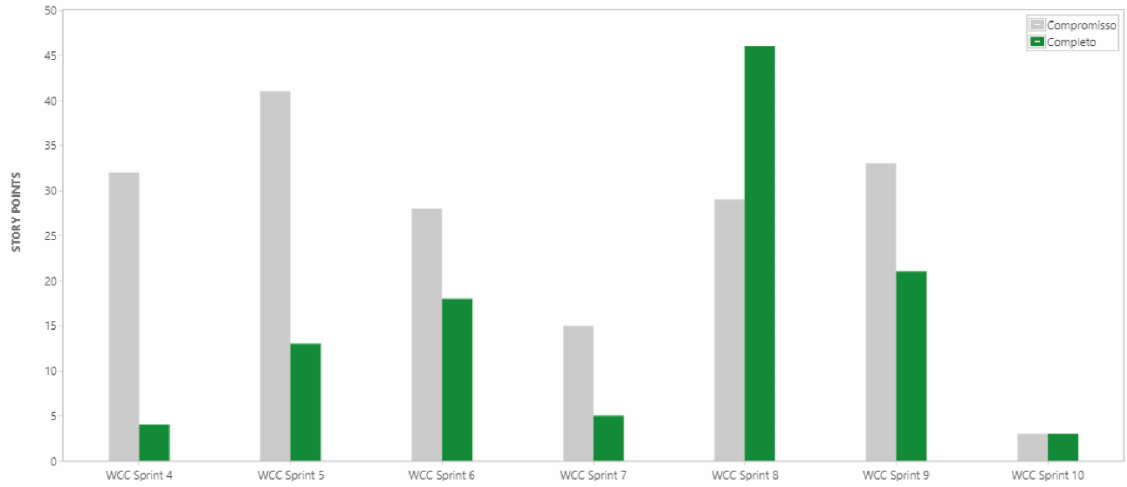

Figura 40 - Gráfico de Barras de Velocity correspondente aos Sprints em estudo
Elaborado pelo Autor

3.4.5 Melhoria Contínua

Nesta fase de aplicação da Framework AMS, estudou-se as visualizações obtidas através da utilização das métricas, a partir do JIRA Software e aplicaram-se as melhorias ao processo de desenvolvimento. As visualizações foram analisadas durante o *Sprint Retropective*, no fim de cada *Sprint*.

Para este caso de estudo a proposta de melhorias foi aplicada no *Sprint Retropective* do Sprint 7, realizando-se a análise antes das alterações de melhoria e consequentemente depois de aplicar as melhorias. Na etapa anterior da aplicação da Framework AMS (Subseção 3.4), é possível analisar os efeitos inerentes às alterações aplicadas e consequentemente verificar as melhorias ligadas ao processo de desenvolvimento.

Senso assim, espera-se que a equipa de desenvolvimento da *We Can Charge* continue a aplicar a filosofia de melhoria contínua no projeto e obter cada vez melhores resultados, ao longo do tempo de desenvolvimento do projeto.

4. CONCLUSÕES

Este capítulo final da dissertação descreve as conclusões retiradas da investigação realizada, comparando os objetivos traçados para este trabalho, com os resultados obtidos no estudo de caso. Nesta fase é realizado um resumo do trabalho desenvolvido, descrito o contributo do estudo realizado, apresentada as limitações que surgiram ao longo do projeto e a proposta de trabalho futuro.

A dissertação desenvolvida partiu da premissa que era possível realizar um estudo de caso, a uma Startup, aplicando uma *framework* de métricas a um projeto ágil e realizar uma proposta de melhoria ao processo de desenvolvimento do projeto. Para criar essa proposta de melhoria ao processo de desenvolvimento da Startup, foi necessário acompanhar a equipa de desenvolvimento e analisar os ciclos *Sprint* de desenvolvimento do produto com a finalidade de obter resultados positivos nos seguintes *Sprints*. De forma a concluir a investigação com sucesso foi delineado um conjunto de objetivos que serviram de referência ao longo da dissertação desenvolvida.

Hoje em dia, as organizações não têm margem para erro quando se fala sobre desenvolvimento de produto. Para que as empresas se mantenham competitivas com o mercado circundante é importante aumentar a exigência de desenvolvimento. Fatores como o custo, a qualidade e os prazos de desenvolvimento do produto são um grande foco das empresas para vingarem no mercado das tecnologias, já que a competitividade é bastante elevada. Deste modo, é crucial manter monitorizado o processo de desenvolvimento de uma empresa, com o objetivo de maximizar o rendimento da equipa de desenvolvimento do produto.

O primeiro objetivo traçado para esta dissertação centrou-se em estabelecer uma parceria com uma Startup, de forma a poder acompanhar o processo de desenvolvimento de um projeto. Sendo assim, foi desenvolvida uma parceria com a Startup *We Can Charge*, uma empresa sediada no edifício do Gnration, em Braga, completando o propósito da dissertação, servindo de "cobaia" para o estudo de caso realizado. Este projeto tem o objetivo de criar uma rede de carregadores de carros elétricos, descentralizado das grandes multinacionais e assim dar poder a qualquer pessoa ou empresa em construir a sua própria rede ou ter o seu próprio carregador.

De seguida, foi realizada uma exaustiva revisão bibliográfica, onde se consolidou os conhecimentos teóricos sobre os temas em estudo nesta dissertação. Dentro da revisão bibliográfica foram abordados temas relacionados com a engenharia de software, gestão de projetos e as metodologias de desenvolvimento, com maior foco no Scrum (metodologia utilizada pelo projeto da *We Can Charge*), métricas utilizadas em projetos ágeis e *frameworks* de métricas desenvolvidas, para usar

como referência, em empresas de desenvolvimento de software. A revisão bibliográfica foi elaborada ao longo do desenvolvimento da investigação, consoante as necessidades de fundamentar o estudo realizado.

Um dos tópicos mais importantes em estudo na dissertação, recaiu sobre as métricas de desenvolvimento de software. Houve a necessidade de realizar um levantamento intensivo de métricas, com o intuito de perceber quais seriam as mais indicadas a usar. A informação é vasta e foi decidido utilizar métricas aplicadas essencialmente em projetos ágeis, como é o caso da *We Can Charge*, focadas no processo de desenvolvimento do produto. Um dos pontos essenciais para selecionar o conjunto de métricas usadas na dissertação, foi utilizar a ferramenta de gestão de projetos da *We Can Charge*, JIRA Software. Ferramenta com grande utilidade na gestão, análise e acompanhamento do projeto.

No seguimento do estudo realizado às métricas usadas no projeto, percebeu-se que *frameworks* de métricas poderiam ser aplicadas no estudo de caso. Infelizmente a informação foi escassa. Foram selecionadas três *frameworks* de métricas para complementar a informação bibliográfica, utilizando no estudo de caso a *Framework AMS*, devido à sua versatilidade e adaptação a vários cenários, com a finalidade de estudar o processo de desenvolvimento da *We Can Charge*. De qualquer forma, todas as *frameworks* poderiam ser aplicadas ao desenvolvimento ágil, mas não se adaptavam da mesma forma ao estudo aqui pretendido realizar.

Na etapa seguinte realizou-se o acompanhamento e análise do processo de desenvolvimento do projeto, através do acesso dado à ferramenta de gestão de projetos, JIRA Software e do acompanhamento semanal das reuniões de *Sprint* da equipa de desenvolvimento. Sem este acesso seria difícil recolher os dados que basearam o caso de estudo. Este foi um processo exaustivo de recolha e comparação de dados, onde foram acompanhados sete ciclos *Sprint* do desenvolvimento do projeto. Com isto, aplicou-se a *framework* desenvolvida a cada um dos *Sprints* e analisado os dados recolhidos a partir da utilização do conjunto de métricas. No primeiro grupo de *Sprints* retirou-se dados importantes (através da aplicação do conjunto de métricas) para apresentar a proposta de melhoria do processo. No segundo grupo de *Sprints* pode-se verificar as melhorias significativamente positivas, resultado da aplicação da proposta de melhorias ao processo.

Após a análise dos primeiros quatro *Sprints* e da aplicação do referencial e do conjunto de métricas aplicou-se a proposta de melhorias ao processo de desenvolvimento. A proposta foi apresentada no seguimento dos dados obtidos no primeiro grupo de *Sprints*, com cooperação do Scrum Master, da *We Can Charge*. Com os dados obtidos foi possível formalizar um conjunto melhorias

ao processo, com o consentimento do Scrum Master, que posteriormente foram aceites e implementadas no seguimento do desenvolvimento do produto. Contudo, as alterações realizadas focam-se essencialmente no processo organizacional da equipa, muito devido à falta de experiência da equipa em usar os mecanismos do Scrum. Como é esperado esta inexperiência revelou-se positiva para o caso de estudo desenvolvido, já que se pode ter uma melhor perceção das alterações realizadas. Este é um dos maiores contributos da dissertação, para a comunidade científica, com o estudo de caso realizado.

De forma geral, os objetivos propostos para a realização desta dissertação foram cumpridos com sucesso. Durante os meses de desenvolvimento da dissertação surgiram limitações naturais ao trabalho desenvolvido tanto ao nível da pesquisa bibliográfica como ao nível do caso de estudo. Uma das limitações sentidas na pesquisa bibliográfica está relacionada com a quantidade de métricas encontradas para analisar o desenvolvimento de um projeto. Quando se procura analisar o processo de desenvolvimento de um projeto torna-se bastante difícil selecionar as métricas a usar relacionadas com a análise do processo, sendo que existe um catálogo vasto de métricas e que preenchem os requisitos a várias categorias de análise ao processo do projeto.

Outra limitação notada no estudo bibliográfico surgiu com a pesquisa de *frameworks de* métricas previamente desenvolvidas, devido à reduzida informação encontrada sobre a matéria. Para este estudo procurava-se uma *framework* já desenvolvida para estandardizar a aplicação das métricas de análise do processo, mostrando-se uma tarefa árdua no estudo de caso. Optou-se por escolher uma *framework* que fosse versátil o suficiente para cumprir com os propósitos da dissertação.

Em relação ao estudo de caso, uma das limitações notadas está relacionado com a proposta de melhoria apresentada. A proposta de melhoria deveria apresentar um número maior de recomendações, mas o estudo teria de ser mais intenso e duradouro e o acesso ao projeto deveria ser desbloqueado totalmente. O mesmo acontece com as métricas utilizadas no estudo de caso. A limitação encontrada na seleção de métricas surgiu também do reduzido acesso à informação do projeto.

No seguimento do estudo de caso realizado para esta dissertação, seria interessante para trabalho futuro, aprofundar e continuar a analisar, apresentando melhorias ao projeto em estudo, já que quando o estudo foi concluído, ainda o projeto estava em processo de desenvolvimento. A atualização e melhoramento da *framework* de métricas selecionadas seria algo a pensar no futuro, para se tornar mais intuitiva, simples e eficaz na sua aplicação, para obtenção de melhores resultados.

Aplicar este referencial de métricas a outros estudos de caso semelhantes, essencialmente a projetos desenvolvidos por Startups e perceber se ajudariam a melhorar o processo de desenvolvimento das mesmas é algo que pode contribuir para o desenvolvimento tecnológico e eficiência das empresas no setor das tecnologias de informação.

As métricas usadas para analisar, acompanhar e apresentar melhorias ao mecanismo do processo de desenvolvimento são fundamentais para o correto funcionamento do desenvolvimento do produto e da gestão do trabalho em equipa, ajudando a detetar os erros e falhas que podem estar a ser cometidos ao longo do projeto.

Por último é interessante reter que o processo de desenvolvimento de um projeto é importantíssimo para o desenvolvimento do produto e a sua qualidade. Uma boa gestão do processo, aplicando sucessivas melhorias ajuda a olear a máquina que é todo este mecanismo e que consequentemente ajuda a equipa a funcionar no limiar do rendimento máximo. Consecutivamente, ajustes podem ser implementados e melhorar a qualidade, custo e tempo de desenvolvimento do produto. Estas três características relacionadas com o produto, podem definir o sucesso ou insucesso de um projeto.

BIBLIOGRAFIA

Ahmad, M. O., Dennehy, D., Conboy, K., & Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. *Journal of Systems and Software*, *137*, 96–113. https://doi.org/10.1016/J.JSS.2017.11.045

Alliance, A. (2017). Agile 101. In *On Agile 2017 Conference*.

Alshamrani, A., & Bahattab, A. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *IJCSI International Journal of Computer Science Issues*.

APM BoK. (2019). Association for Project Management Body of Knowledge. In *APM Body Of Knowledge*.

Ashbacher, C. (2010). Succeeding With Agile: Software Development Using Scrum, by Mike Cohn. *The Journal of Object Technology*. https://doi.org/10.5381/jot.2010.9.4.r1

Atlassian. (n.d.). *What is Agile?* Retrieved January 5, 2022, from https://www.atlassian.com/agile

Azure DevOps. (2021). *What is Scrum?* . Retrieved May 10, 2021, from https://docs.microsoft.com/en-us/devops/plan/what-is-scrum

Baxter Pamela, & Jack, S. (1990). Qualitative case study methodology: study design and implementation for novice researchers. *The Qualitative Report*.

Beck, K, Beedle, M., Bennekum, A. Van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*. The Agile Alliance.

Beck, Kent, Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Agile Manifesto*. Software Development.

Benavides, V. (n.d.). *KPIs to Measure Traditional and Agile Project Delivery*. Retrieved June 25, 2021, from https://aspirent.com/kpis-to-measure-traditional-and-agile-project-delivery/

Bourque, P., & Fairley, R. E. (2014). Software Engineering - Body of Knowledge. In *IEEE Computer Society*.

Cabri, A., & Griffiths, M. (2006). Earned value and Agile reporting. *Proceedings - AGILE Conference, 2006*. https://doi.org/10.1109/AGILE.2006.21

Calazans, A. T. S., & Alvarenga, M. S. (2014). Métricas para Métodos Ágeis de Desenvolvimento – Um Estudo Comparativo. In *XI Simpósio de Gestão e Tecnologia*.

Canedo, E. D., & Da Costa, R. P. (2018). Methods and metrics for estimating and planning agile software projects. *Americas Conference on Information Systems 2018: Digital Disruption, AMCIS 2018*.

Chowdhury, A. (2021). *Agile Metrics: The 15 That Actually Matter for Success - Plutora.com*. Retrieved May 10, 2021, from https://www.plutora.com/blog/agile-metrics

Cohen, D., Lindvall, M., & Costa, P. (2003). A State of the Art Report: Agile Software Development. *DACS SOAR Report*.

Correia, L. (2018). *Desenvolvimento de uma framework de métricas para projetos ágeis*.

Dash, S. N. (2020). *Earned Value Management (EVM) in Agile Development | MPUG*. Retrieved May 10, 2021, from https://www.mpug.com/earned-value-management-evm-in-agile-development/

DeMarco, T., & Lister, T. (2014). Peopleware : Productive Projects and Teams, 3rd Ed. In *Journal of Chemical Information and Modeling*.

Douglas Neves, C. (2020). *Métricas em Processos Ágeis: Um Framework Baseado em Storytelling para Visualizações*.

Drebes Pedron, C. (2008). *O método de Investigação - Caso de Estudo*.

Elgabry, O. (2017). *Software Engineering — Software Process and Software Process Models (Part 2) | by Omar Elgabry | OmarElgabry's Blog | Medium*. Retrieved March 19, 2021, from https://medium.com/omarelgabrys-blog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc

Estdale, J., & Georgiadou, E. (2018). Applying the ISO/IEC 25010 Quality Models to Software Product. *Communications in Computer and Information Science*, *896*. https://doi.org/10.1007/978-3-319-97925-0_42

Geckoboard, T. (2019). *A complete guide to key performance indicators: Definition, uses and examples*. Retrieved March 2, 2021, from https://www.geckoboard.com/blog/what-is-a-key-performance-indicator-kpi/

Gustafsson, J., Supervisor, J. 2011, & Feldt, R. (2011). *Model of Agile Software Measurement: A Case Study*.

Harold Kerzner, P. d. (2010). Project Management, A Systems Approach To Planning, Scheduling, and Controlling. In *Patent Project Management*.

Hayes, W., Miller, S., Lapham, M. A., Wrubel, E., & Chick, T. (2014). *Agile Metrics: Progress Monitoring*

of Agile Contractors. http://www.sei.cmu.edu

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. In *Computer*. https://doi.org/10.1109/2.947100

Hoon Kwak, Y. (2003). Brief History of Project Management. In *The Story of Managing Projects*.

Huang, C. C., & Kusiak, A. (1996). Overview of kanban systems. *International Journal of Computer Integrated Manufacturing*. https://doi.org/10.1080/095119296131643

International Organization For Standardization. (2011). Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models. *ISO/IEC 25010:2011*, *2*(Resolution 937).

ISO. (2018). *ISO 21508:2018 - Earned Value Management in Project and Programme Management*. www.sis.se.buytheentirestandardviahttps://www.sis.se/std-80003813

ISO 25010. (n.d.). Retrieved April 22, 2021, from https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?start=0

JJ Sutherland. (2018). *The 3-5-3 Structure of Scrum*. Retrieved March 11, 2021, from https://www.scruminc.com/the-3-5-3-of-scrum/

Kanbanize. (n.d.-a). *O Quadro Kanban Explicado em 5 Passos Simples*. Retrieved January 5, 2022, from https://kanbanize.com/pt/recursos-kanban/primeiros-passos/o-que-e-quadro-kanban

Kanbanize. (n.d.-b). *O que é Kanban? Definição e Detalhes Explicados*. Retrieved October 8, 2021, from https://kanbanize.com/pt/recursos-kanban/primeiros-passos/o-que-e-kanban

Kanbanize. (2019). *What is a WIP Limit and how to set it up.* Retrieved March 20, 2021, from https://www.kanbanize.com

Khan, Z. (2014). *Scrumban-Adaptive Agile Development Process: Using scrumban to improve software development process*. *May*, 122. https://www.theseus.fi/bitstream/handle/10024/77014/Khan_Zahoor.pdf%0Ahttp://theseus32-kk.lib.helsinki.fi/handle/10024/77014

Knaflic, C. N. (2015). Storytelling with data : a data visualization guide for business professionals / Cole Nussbaumer Knaflic. In *Storytelling with data : a data visualization guide for business professionals*.

Kniberg, H., & Skarin, M. (2010). Kanban and Scrum-making the most of both. In *Work*.

Kumar, C. S., & Panneerselvam, R. (2007). Literature review of JIT-KANBAN system. *International Journal of Advanced Manufacturing Technology*. https://doi.org/10.1007/s00170-005-0340-2

Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2014). Why are industrial agile teams using metrics and

how do they use them? *5th International Workshop on Emerging Trends in Software Metrics, WETSoM 2014 - Proceedings*. https://doi.org/10.1145/2593868.2593873

Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using metrics in Agile and Lean software development - A systematic literature review of industrial studies. In *Information and Software Technology* (Vol. 62, Issue 1). https://doi.org/10.1016/j.infsof.2015.02.005

Ladas, C. (2009). Scrumban: essays on kanban systems for lean software development. In *Modus Cooperandi Press*.

Lynn, R. (n.d.). *Lean Metrics to Improve Flow*. Retrieved October 21, 2021, from https://www.planview.com/resources/guide/what-is-agile-program-management/lean-metrics-improve-flow/

Mariana Racasan. (2021). *What Is a Sprint Backlog and What to Include in One*. Retrieved October 15, 2021, from https://blog.zenhub.com/what-is-a-sprint-backlog/

Miles, M., & Huberman, M. (1994). Data management and analysis methods. *Handbook of Qualitative Research*.

Moniruzzaman, M., Syed, D., & Hossain, A. (2013). *Comparative Study on Agile software development methodologies*.

O'Hara, Susan; Levin, G. (2000). *Using metrics to demonstrate the value of project management*. Project Management Institute Annual Seminars & Symposium. Retrieved October 2, 2021, from https://www.pmi.org/learning/library/metrics-demonstrate-value-project-management-485

Padmini, K. V. J., Dilum Bandara, H. M. N., & Perera, I. (2015). Use of software metrics in agile software development process. *MERCon 2015 - Moratuwa Engineering Research Conference*, 312–317. https://doi.org/10.1109/MERCon.2015.7112365

Pahuja, S. (2015). *What is Scrumban?* Retrieved June 10, 2021, from Agile Alliance.

Paré, G. (2004). Investigating Information Systems with Positivist Case Research. *Communications of the Association for Information Systems*. https://doi.org/10.17705/1cais.01318

Pavel. (2021). *Kanban Metrics: Throughput | Hygger University*. Retrieved March 10, 2021, from https://university.hygger.io/en/articles/3119280-kanban-metrics-throughput

Pegoraro, R. A. (2014). *Métricas de Avaliação para Abordagens Ágeis em projetos de software*.

Project Management Institute. (2008). A guide to the project management body of knowledge (PMBOK® guide). In *Project Management Journal*.

Radigan, D. (n.d.). *Five agile metrics you won't hate | Atlassian*. Retrieved June 19, 2021, from https://www.atlassian.com/agile/project-management/metrics

Raphael Batagini. (2019). *Kanban e suas Principais Métricas* . Retrieved March 12, 2021, from https://medium.com/@raphaelbatagini/kanban-e-suas-principais-métricas-228d938326fb

Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques | BibSonomy. *Proce IEEE WESTCON, Los Angeles*.

Schwaber, K. (1997). SCRUM Development Process. In *Business Object Design and Implementation*. https://doi.org/10.1007/978-1-4471-0947-1_11

Schwaber, K., & Sutherland, J. (2017). The Scrum Guide: The Definitive The Rules of the Game. *Scrum.Org and ScrumInc*.

Scrum.org. (n.d.). *Scrum Values Poster*. Retrieved January 4, 2022, from https://www.scrum.org/resources/scrum-values-poster

Soares, M. D. S. (2003). Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. *INFOCOMP Journal of Computer Science*.

Software Engineering Course (SWEBOK) | IEEE Computer Society. (n.d.). Retrieved October 7, 2021, from https://www.computer.org/education/bodies-of-knowledge/software-engineering

Sommerville, I. (2006). Software Engineering (8th Ed.). In *Software Engineering (8th Ed.)*.

Sugimori, Y., Kusunoki, K., Cho, F., & Uchikawa, S. (1977). Toyota production system and kanban system materialization of just-in-time and respect-for-human system. *International Journal of Production Research*. https://doi.org/10.1080/00207547708943149

Sulaiman, T., Barton, B., & Blackburn, T. (2006). AgileEVM - Earned value management in scrum projects. *Proceedings - AGILE Conference, 2006*. https://doi.org/10.1109/AGILE.2006.15

Sutherland, J. (2010). Jeff Sutherland's Scrum handbook. *Scrum Training Institute*.

Tereso, A., Ribeiro, P., Fernandes, G., Loureiro, I., & Ferreira, M. (2019). *Project Management Practices in Private Organizations*. https://doi.org/10.1177/8756972818810966

The new new product development game. (1986). *Journal of Product Innovation Management*. https://doi.org/10.1016/0737-6782(86)90053-6

Willaert, S. S. A., De Graaf, R., & Minderhoud, S. (1998). Collaborative engineering: A case study of Concurrent Engineering in a wider context. *Journal of Engineering and Technology Management - JET-M*. https://doi.org/10.1016/S0923-4748(97)00026-X

Wysocki, R. K. (2014). Effective Project Management: Tradicional, Agile, Extreme. In *John Wiley & Sons, Inc.*

Yin, R.K. (2003). Case study methodology R.K. Yin (2003, 3rd edition). Case Study Research design and methods. Sage, Thousand Oaks (CA)..pdf. In *Case Study Research: design and methods*.

Yin, Robert K. (1998). The Abridged Version of Case Study Research. In *Handbook of applied social research methods*.

Yin, Robert K. (2002). Case Study Reserach - Design and Methods (Second Edition. In *Sage Publications*.

YOUNG, J. (2020). *Metrics Definition*. Retrieved March 15, 2021, from https://www.investopedia.com/terms/m/metrics.asp

Yuval Yeret. (2012). *So what is Scrumban?* Retrieved March 13, 2021, from http://yuvalyeret.com/so-what-is-scrumban/