Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

Aplicação de uma metodologia ágil de desenvolvimento de software numa organização do sector público

Afonso Fernandes Portela Ribeiro

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Informática e Gestão

Orientadora:
Professora Luísa Domingues,
ISCTE-IUL

Outubro, 2018

Agradecimentos

No momento em que termina mais um ciclo do meu percurso académico, não posso deixar de fazer os devidos agradecimentos às pessoas e instituições que assim o permitiram.

Em primeiro lugar à Professora Doutora Luísa Domingues por me ter apresentado o desafio e me ter dado a oportunidade de explorá-lo e desenvolve-lo sobre a sua orientação.

Um especial agradecimento à Sara Moras que me motivou e apoiou nos momentos de maior dificuldade.

Aos meus pais e à minha família por terem criado todas as condições necessárias para a minha formação académica e pessoal.

Ao IGFEJ que me deu oportunidade de aplicar todos os conhecimentos adquiridos num contexto real permitindo, dessa maneira, a obtenção de resultados práticos no contexto da utilização das metodologias ágeis.

Por último gostaria de agradecer à instituição ISCTE por me ter proporcionado a oportunidade de aprender e crescer não só academicamente como pessoalmente, colocando sempre o bem-estar dos alunos em primeiro lugar permitindo a melhor experiência de aprendizagem possível.

Resumo

As metodologias de desenvolvimento de *software* têm sofrido um crescimento e uma consequente maturação ao longo dos últimos anos. Os métodos mais antigos, denominados tradicionais, têm sido substituídos pelas novas práticas ágeis, existindo estudos empíricos que o comprovam. No entanto, a falta de informação na aplicação destas novas metodologias no sector público levam à seguinte questão: Em que condições é que as metodologias ágeis são uma opção viável no desenvolvimento de *software* no sector público?

Este tipo de organizações tem uma cultura organizacional e um *modus operandi* muito próprio que podem tornar a implementação destas metodologias de desenvolvimento de *software* um desafio.

O objetivo da presente dissertação passa por perceber os impactos da aplicação de uma metodologia ágil de desenvolvimento de software numa empresa do sector público.

No sentido de mitigar a resistência dos colaboradores às novas práticas e maximizar a sua adequação no contexto em questão, a implementação da metodologia será suportada por duas fases: 1ª Metodologia ágil a ser implementada na organização que será criada com base na revisão literária, no contexto organizacional da empresa neste sector, e no *feedback* dos colaboradores; 2ª Estratégias de gestão da mudança que terão como fundamento a revisão literária e casos de estudo.

O desenvolvimento da dissertação contém a justificação e criação das fases acima referidas, bem como a análise dos resultados da implementação da metodologia num projeto real.

Palavras-Chave: Gestão da mudança; Gestão de projetos; Metodologias ágeis; Sector público.

Abstract

Software development methodology have been experiencing growth and consequential maturation during the last years. Ancient methods, denominated traditional, have been replaced by new agile practices, according to empirical studies that prove it. However, the lack of information for appliance of these new methodologies in the public sector brings us this question : In which conditions are these agile methodologies a viable option in the development of software in the public sector ?

This kind of organizations work under a very unique culture/*modus operandi* which can make the implementation of these software development methodologies chalenging.

The main goal of this dissertation is to focus in understanding the impacts of the application of a agile software development methodology in a public sector company.

In order to mitigate reluctance from colaborators to applicate these new practices, their implementation shall be supported by two phases: 1º Agile methodology implemented in organization which will be created based in literary revision, company organizationl context in this sector and in collaborator feedback. 2º Change management strategies substantiated with literary revision and study cases.

This dissertation development contains justification and phase creation for what is referred above, as well as result analyses of implementation of this methodology in a real/concrete project

Keywords: Change Management; Project Management; Agile Methodologies; Public Sector.

Índice

Índice de Tabelas

Índice de Figuras

Lista de Abreviaturas e Siglas

SI – Sistemas de informação

TI – Tecnologias de informação

XP – *Extreme Programming*

FDD – *Feature Driven Development*

ASD – *Adaptative Software Development*

DSDM – *Dynamic System Development Method*

RACI – *Responsible, Accountable, Consulted e Informed*

PMBOK – *Project Management Body of Knowledge*

IGFEJ -Instituto de Gestão Financeira e Equipamentos da Justiça

Capítulo 1 – Introdução

1.1. Enquadramento do tema

A informatização dos sistemas de informação é um tema presente na vida de todas as organizações que ao longo dos anos têm vindo a atualizar e a melhorar os seus sistemas de informação.

Paralelamente à evolução tecnológica dos sistemas, identifica-se claramente uma revolução no âmbito das metodologias de desenvolvimento de *software*, que se traduzem em práticas para implementação do produto pretendido.

Estas novas metodologias denominam-se ágeis e diferem das tradicionais em alguns pontos que são discutidos no capítulo 2 do presente documento.

De acordo com a revisão literária efetuada no âmbito da presente dissertação, apesar da adesão destas novas metodologias no mercado ser grande, no sector público isso não se verifica. Neste sector a adesão tem sido mais lenta, já que este tipo de organizações culturalmente apresenta-se como mais tradicionais comparativamente com os outros sectores.

A presente dissertação é desenvolvida no IGFEJ (Instituto de Gestão Financeira e Equipamentos da Justiça), que serve como ponto de partida e como caso de estudo para o desenvolvimento e aplicação dos conceitos a explorar. O IGFEJ é uma organização pública portuguesa responsável pelos Sistemas de Informação da Justiça. É uma entidade hierarquizada e dividida em departamentos, dois dos quais estão ligados aos Sistemas de Informação. Um é responsável pela arquitetura e manutenção de sistemas, e outro pelos serviços de suporte tecnológico.

Todos os anos existem vários projetos que são desenvolvidos e monitorizados pelo IGFEJ, cujo âmbito incide quer na manutenção e correção das plataformas, como no desenvolvimento de novas funcionalidades.

O IGFEJ tem acompanhado a evolução tecnológica, revolucionando e informatizando os processos na área da justiça. Apesar deste progresso a nível tecnológico ter agregado vantagens, têm surgido problemas internos no controlo e integração dos novos projetos. Estes problemas devem-se ao facto de os sistemas desenvolvidos serem mais complexos e a dependência entre os mesmos ser maior. Esta dependência induz a que os prazos de entrega sejam menos flexíveis e, consequentemente, a margem para errar seja menor.

Tendo em consideração,

- os novos desafios que se impõem a nível tecnológico,
- os constrangimentos associados aos métodos de desenvolvimento de *software* no setor público,
- a introdução de *know how* atualizado relativamente a diferentes metodologias de desenvolvimento de *software*,

é útil debater/investigar de que forma estas últimas podem ser um suporte positivo nas organizações, ajudando a mitigar, contornar e minimizar os problemas identificados nas organizações dentro deste âmbito. O impacto expectável, apesar de positivo, acarreta desafios acrescidos, evidenciando a necessidade de adaptação (das técnicas/métodos a implementar) às características, comportamentos e dinâmicas específicas do setor em questão.

1.2. Motivação e relevância do tema

A dependência nos sistemas de informação, por parte das organizações, é cada vez maior. Mesmo que o *core business* das entidades não esteja diretamente relacionado com as TI, em algum ponto nos processos da empresa, existe alguma relação/dependência pelos SI. Este fenómeno ocorre não só no sector privado (com um nível de competitividade maior) mas também no sector público.

No sentido de acompanhar as exigências do mercado, as metodologias de desenvolvimento de *software* também se adaptaram, tornando-se mais flexíveis. Apesar destas novas metodologias terem sido adotadas em grande escala pelas entidades privadas, nas públicas isso já não se verificou. Conboy fala sobre os desafios da adoção das práticas ágeis e solicita mais investigação nos efeitos da aplicação das mesmas (Conboy & Wang, 2007). Apesar deste artigo de Conboy datar de 2007, existem outros autores com estudos mais recentes que referem a escassez de documentação nesta área (Kaczorowska, 2015; Karaj & Little, 2013; Powner, 2012).

A pertinência desta dissertação prende-se exatamente com este aspeto. A premissa de que é necessário maior contributo para o conhecimento científico nesta área. Pretende-se que seja documentada a informação relativa aos impactos da adoção de uma metodologia ágil de desenvolvimento de *software* no contexto de uma entidade pública.

1.3. Questões e objetivos de investigação

Verificou-se no subcapítulo 1.2, relativo à motivação e relevância do tema da presente dissertação, que é fundamental o contributo para a literatura científica na área das metodologias ágeis aplicadas nas organizações públicas. Nesse sentido, pretende-se responder a duas questões:

1ª - De que modo é que as metodologias ágeis de desenvolvimento de *software* se adequam ao contexto do desenvolvimento efetuado no sector público e quais os seus impactos?

2ª - Em que condições é que as metodologias ágeis são uma opção viável no desenvolvimento de *software* no sector público?

Para tal, o objetivo geral desta dissertação passa pela aplicação de uma metodologia ágil de desenvolvimento de *software* no IGFEJ e análise dos seus impactos na organização. No sentido de atingir o objetivo definido acima, foram enunciados dois objetivos específicos. O primeiro refere-se à criação de uma metodologia ágil que permita melhorar o desenvolvimento de *software* no IGFEJ reduzindo assim, os problemas no desenvolvimento de software apresentados no capítulo 3. O segundo refere-se à criação de um conjunto de estratégias de gestão da mudança que mitiguem a resistência à adoção da metodologia proposta.

A concretização destes objetivos passa pela interação e recolha de *feedback* dos colaboradores, análise dos processos atuais do IGFEJ, bem como a identificação dos impactos no desenvolvimento de *software* da empresa com a estratégia atualmente utilizada.

Os objetivos serão validados com a implementação, num projeto real no IGFEJ, da metodologia desenvolvida no âmbito desta dissertação, e com a implementação das estratégias de gestão da mudança criadas. A validação por pares também será um componente que será incluída através da publicação de artigo científico na revista *Procedia Computer Science* editada pela *Elsevier*. O artigo tem como tema "*Acceptance of an agile methodology in the public sector*" ("Aceitação de uma metodologia ágil no sector público") e será publicado na 10ª edição da conferência *CENTERIS* (Ribeiro & Domingues 2018).

1.4. Abordagem metodológica

Relativamente à abordagem metodológica, considera-se que a presente dissertação se enquadra no modelo de investigação da *Design Science Research*, na medida em que o objetivo anteriormente definido passa pela criação e validação de um artefacto (denomina-se artefacto à(s) solução(ões) para a classe de problemas levantada). No contexto da presente dissertação, o artefato apresenta-se como a metodologia ágil proposta que procura colmatar os problemas identificados no desenvolvimento de *software* no IGFEJ.

A metodologia de investigação da *Design Science Research* identifica cinco etapas do processo de investigação (Manson 2006):

1. Identificação do problema. Neste ponto pretende-se evidenciar a problemática que se propõe resolver e explicar qual a relação com o artefacto;

2. Sugestão da solução. A seguinte fase pretende demonstrar de que maneira o artefacto foi construído e de que forma o mesmo irá colmatar os problemas identificados na fase anterior;

3. Desenvolvimento das ações necessárias para efetivação da sugestão. Justificar os mecanismos para o desenvolvimento do artefacto bem como a criação do mesmo;

4. Avaliação do artefacto. Nesta fase deve ser clarificado como será validado o artefacto e evidenciar os resultados da aplicação do mesmo, determinando o que ocorreu com sucesso e insucesso;

5. Conclusão. Neste ponto devem ser sintetizadas as principais aprendizagens levadas a cabo em cada fase, referindo o contributo para a comunidade científica.

Para que se tenha um melhor entendimento da maneira como a metodologia de investigação se correlaciona com a estrutura da presente dissertação, de seguida, ir-se-á descrever o que foi efetuado no âmbito da mesma em cada uma das cinco etapas anteriormente evidenciadas:

Na primeira etapa de 'Identificação do problema' foi efetuada uma análise exaustiva do contexto do IGFEJ com maior incidência no processo de desenvolvimento de *software* que atualmente a organização detém. Desta maneira, é possível identificar os vários tipos de projetos elaborados na organização, bem como os problemas inerentes aos mesmos.

De seguida, foram elaboradas as etapas ‘Sugestão da solução’ e ‘Desenvolvimento das acções necessárias para a efetivação da sugestão’. Nestes pontos foi construído o modelo de suporte à implementação de uma metodologia ágil de desenvolvimento de *software*. Foi também justificada a sua criação e de que maneira o mesmo procura colmatar os problemas no desenvolvimento de *software* efetuado no IGFEJ.

Na etapa de ‘Avaliação do artefacto’ foi descrita de que maneira o artefacto foi validado. Nesse sentido são apresentadas três formas de validação. 1ª Avaliação por pares, através da publicação de um artigo científico na revista relativo à aceitação do artefacto no contexto de uma empresa do sector público; 2ª Avaliação do IGFEJ, tendo sido recolhido *feedback* dos colaboradores relativamente à adequação do modelo ao contexto da organização; 3ª Recolha de resultados, através da implementação do modelo num projeto real no desenvolvimento de software no IGFEJ.

Na última etapa da metodologia, etapa da Conclusão, foram sintetizadas as principais ideias resultantes do progresso da presente dissertação, bem como trabalho futuro a desenvolver.

No sentido de complementar e fundamentar as decisões tomadas, nas fases de ‘Sugestão da solução’ e ‘Desenvolvimento das ações necessárias para efetivação da sugestão’, foi efetuada uma revisão literária, apresentada no Capítulo 2. A mesma teve como base os seguintes critérios de seleção, de forma a tornar a pesquisa o mais precisa e com a informação mais relevante possível:

- Deu-se preferência na recolha de informação às fontes de dados mais conceituadas nesta área. Com maior preponderância, destacam-se a ACM, IEEE Xplore, Elsevier, Spring e Wiley.
- Como segundo critério foram selecionados os artigos com maior número de citações tendo em conta o ano em que foram publicados, isto é, com maior impacto na investigação.
- Foi também dada preferência aos autores que mais se destacaram na elaboração de artigos na área das metodologias ágeis e tradicionais.

1.5. Estrutura e organização

A presente dissertação está organizada em sete (7) capítulos.

O primeiro capítulo integra a Introdução, os Objetivos, e a Metodologia de Investigação utilizada no desenvolvimento da presente dissertação.

O segundo capítulo contém toda a parte da revisão literária e divide-se em cinco (5) subcapítulos:

- O 1º subcapítulo começa com a definição e explicação das metodologias ágeis e a diferença das mesmas para as tradicionais.
- No 2º subcapítulo são aprofundadas as metodologias ágeis mais utilizadas em contexto real, de forma a perceber que práticas são mais aplicadas e de que maneira são implementadas.
- Posteriormente é analisado o paradigma *Lean* e a relação do mesmo com as metodologias ágeis.
- Foram também clarificados os principais problemas na implementação das metodologias ágeis nas entidades públicas no sentido de perceber, neste contexto, que tipo de informação e análises já foram recolhidas sobre este tema e que semelhanças existem com o tópico da dissertação.

O terceiro capítulo passa pela contextualização da organização onde será efetuado o estudo empírico do âmbito desta dissertação.

O quarto capítulo integra a definição e justificação da metodologia ágil de desenvolvimento de software proposta, bem como das estratégias de gestão da mudança (componentes do modelo).

No quinto capítulo serão extraídos os resultados da aplicação das componentes do modelo criado no capítulo anterior num projeto do IGFEJ, e será efetuada a análise crítica dos mesmos.

No sexto capítulo, serão extraídas as principais conclusões da dissertação e serão enunciados o trabalho futuro e as sugestões de melhoria.

Por último, no sétimo capítulo é efetuado uma breve descrição dos contributos do presente trabalho para a comunidade científica.

Capítulo 2 – Revisão da Literatura

2.1 Metodologias ágeis

As metodologias ágeis caracterizam-se por trabalho colaborativo, competências multidisciplinares, descentralização das decisões, envolvimento do cliente, e equipas preferencialmente pequenas. Já os métodos tradicionais focam-se no desenvolvimento individual, competências especializadas, decisões administrativas, pouco envolvimento do cliente, e equipas geralmente grandes (Madadipouy, 2015). As metodologias tradicionais identificam-se também, como tendo um planeamento extensivo e processos rígidos (Boehm, 2002).

As metodologias ágeis foram criadas para mitigar as limitações/fragilidades dos métodos tradicionais de desenvolvimento de *software*, como o excesso de documentação e a ineficácia do "*time to market*" (Jalali, Wohlin, & Angelis, 2014). Surgiram no fundo, para dar resposta à ineficiência dos métodos de desenvolvimento de *software* existentes, devido à rápida mudança dos contextos envolventes (Highsmith, 2002).

O desenvolvimento de *software* tradicional é considerado inflexível e falha na resposta aos pedidos de mudança dos clientes. Pelo contrário, as metodologias de desenvolvimento ágil contêm práticas que permitem a rápida adaptação às necessidades do produto (Papadopoulos, 2015).

As metodologias ágeis surgem num período em que a exigência por parte dos clientes relativa aos produtos e serviços contratados sofrem uma mudança. Esta renovação é, entre outras razões, consequência do aumento da competitividade empresarial. Este fenómeno levou a que os clientes começassem a dar mais valor a aspetos como a rapidez da entrega e a redução dos custos do desenvolvimento de *software* (Dybå & Dingsøyr, 2008). Para além desta mudança no valor percebido dos clientes no desenvolvimento de *software*, as empresas fornecedoras pretendiam obter mais lucro com menos recursos, com o objetivo de efetuar mais projetos num período de tempo mais reduzido (Madadipouy, 2015). As metodologias ágeis surgem para colmatar estas necessidades levantadas por ambas as partes.

O conceito das metodologias ágeis obtém visibilidade em Fevereiro de 2001, quando é elaborado e publicado o manifesto para o desenvolvimento ágil de *software* ("What is Agile Software Development? | Agile Alliance," n.d.). Apesar de os princípios e práticas

do desenvolvimento ágil de *software* não serem completamente novos para a comunidade de *software*, o modo como foram articulados e integrados numa *framework* teórica ou prática certamente foi (Williams & Cockburn, 2003). Após a criação do manifesto, foi constituída a *Agile Alliance* no sentido de incentivar os investigadores a explorarem e partilharem ideias e experiências sobre a metodologia ("What is Agile Software Development? | Agile Alliance," n.d.). Tal como é recorrente em novas tecnologias e teorias emergentes no desenvolvimento de *software*, a prática da metodologia ágil tem proliferado devido à investigação, promoção e disseminação da comunidade de investigadores e consultores (Conboy, 2009).

Os métodos ágeis têm como objetivo atingir a satisfação dos clientes, entregando de maneira contínua *software* de valor (Madadipouy, 2015). Ao longo do tempo, investigadores desta área contribuíram para este tema, no sentido de clarificar a definição e explicação da metodologia ágil.

Para Henderson-Sellers & Serour (2005) agilidade implica a capacidade de rapidamente, e de forma flexível, responder às mudanças do negócio e dos aspetos técnicos. Lee & Xia (2010) define desenvolvimento ágil de *software* como a presença de equipas com espaço para eficientemente e eficazmente responder e incorporar os pedidos de mudança de requisitos durante o ciclo de vida do projeto.

Segundo os princípios de desenvolvimento ágil de *software* definidos no manifesto de 2001, a criação de valor e satisfação do cliente é feita através da precoce e contínua entrega de *software* e da possibilidade de alteração dos requisitos mesmo na fase final do projeto. Os princípios também referem a importância das equipas terem autonomia na sua organização, de maneira a potenciar a sua produtividade e criatividade, bem como a importância do envolvimento ativo dos clientes na fase de desenvolvimento do projeto para que seja mais fácil a partilha de *feedback*. Os princípios de desenvolvimento ágil de *software* definidos no manifesto de 2001 são descritos abaixo:

- A maior prioridade é satisfazer o cliente através da entrega atempada e contínua de *software* de qualidade;
- A mudança de requisitos, mesmo no final do desenvolvimento, é bem-vinda;
- Entregar frequentemente *software* funcional. Desde duas em duas semanas até de dois em dois meses, com preferência para períodos mais pequenos;
- Pessoas do negócio e do desenvolvimento têm de trabalhar diariamente em conjunto ao longo de todo o projeto;

- Construir projetos à volta de pessoas motivadas. Dar-lhes o ambiente e suporte necessário e acreditar neles para efetuar o trabalho;
- O método mais eficiente e eficaz para entregar informação, seja à equipa ou entre a equipa, é através de conversas presenciais;
- A medida primária de progresso do projeto é com *software* que funcione;
- Processos ágeis promovem um desenvolvimento sustentável. Os patrocinadores, programadores e utilizadores, devem ser capazes de manter um ritmo constante indefinidamente;
- A atenção contínua na excelência técnica e um bom *design*, aumentam a agilidade do projeto;
- Simplicidade, como a arte de maximizar a quantidade de trabalho por fazer, é essencial;
- A melhor arquitetura, requisitos e *design* surgem de equipas que têm autonomia para se organizarem sozinhas;
- Num intervalo regular, a equipa reflete sobre como se tornar mais eficaz, ajustando e aperfeiçoando os processos em conformidade.

O manifesto de 2001 contemplou também 4 valores: Pessoas e Interações (acima de processos e ferramentas), *software* a funcionar (acima de documentação clara/responsiva), colaboração do cliente acima de negociações contratuais, e resposta à mudança em detrimento do plano inicial.

Em termos percentuais o número de projetos finalizados com sucesso, utilizando metodologias ágeis, é superior aos projetos elaborados com metodologias tradicionais. Um artigo de 2017 aponta que o desenvolvimento de projetos com metodologias ágeis o índice de sucesso é 28% acima que os desenvolvidos com metodologias tradicionais (Ganesh Jonnalagadda & Sarah Shafey, 2017). Outro estudo, sintetizado na Tabela 1, que teve como fundamento cerca de 10.000 projetos de desenvolvimento de *software*, verificou, em 2015, que 39% dos projetos desenvolvidos com metodologias ágeis são bem sucedidos, enquanto que os projetos tradicionais são bem sucedidos apenas em 11% (Standish Group 2015).

Tabela 1 – Percentagem de sucesso em projetos ágeis vs tradicionais

Método	Bem sucedido	Desafiante	Falhado
Ágil	39%	52%	9%
Tradicional	11%	60%	29%

Fonte: (Ganesh Jonnalagadda & Sarah Shafey, 2017)

Bem sucedido - Um projeto bem sucedido cumpre as três dimensões do projeto: custo, âmbito e *deadline;*

Desafiante - Um projeto desafiante cumpre pelo menos duas das três dimensões do projeto: custo, âmbito e *deadline*;

Falhado - Um projeto falhado é um projeto que é cancelado antes de terminado ou que foi concluído mas não foi utilizado.

Apesar de a taxa de sucesso nos projetos ágeis ser elevada comparativamente com as metodologias tradicionais, foram encontrados desafios relativos à implementação das metodologias ágeis nas empresas. Os principais desafios apresentam-se na Tabela 2.

Tabela 2 – Desafios na implementação das metodologias ágeis

Desafios	Referências
Adequação do Projeto:	(Lopez-Martinez, Juarez-Ramirez, Huertas, Jimenez, & Guerra-Garcia, 2016) (Hajjdiab & Al Shaima Taleb, 2011) (Mahanti, 2006) (Abdalhamid & Mishra, 2017)
Existe uma dificuldade de escalar a metodologia ágil a projetos de maior dimensão	
A ausência de um Projeto Piloto na transição do método tradicional para o método SCRUM, de modo a avaliar como as equipas se adaptam	
Aspetos organizacionais:	(Hajjdiab & Al Shaima Taleb, 2011)

Existe falta de capacidade para alterar a cultura organizacional, o que inclui a ausência de suporte dos gestores de topo das entidades	(Lopez-Martinez et al., 2016) (Mahanti, 2006) (Abdalhamid & Mishra, 2017) (Inayat, Salim, Marczak, Daneva, & Shamshirband, 2015)
Tentativas de balancear os métodos tradicionais com os novos métodos, devido à existência de pressões para que se utilizem as práticas tradicionais	
Constrangimentos financeiros que justificam a não contratação de recursos (e.g. Agile Master)	
Limitações dos recursos na adoção das metodologias ágeis:	(Hajjdiab & Al Shaima Taleb, 2011) (Lopez-Martinez et al., 2016) (Abdalhamid & Mishra, 2017) (Hamed & Abushama, 2013) (Kiran & V Rama, 2013) (Inayat et al., 2015)
Falhas de comunicação entre o Product Owner e o cliente, falta de experiência com o novo método	
Formação disfuncional e inadequada	
Existência de projetos que necessitam de esforço diário dos membros da equipa, levando a que os mesmos não tenham disponibilidade para comparecer nas diferentes reuniões	
Resistência à mudança	(Hajjdiab & Al Shaima Taleb, 2011) (Lopez-Martinez et al., 2016) (Mahanti, 2006) (Abdalhamid & Mishra, 2017)
Dificuldade em convencer a equipa de topo a investir num novo método	
O modo de abordar o projeto, para muitas equipas, está já muito intrínseco e é difícil de mudar. É necessário fornecer formação adequada, de forma a que se compreendam os princípios das práticas ágil	
Existência de mentes mais fechadas para a entrada de novas práticas	

Membros da equipa com algum poder podem ter receio que o mesmo se dilua ao ter de partilhar informações com colegas mais juniores	

2.2. Diferentes tipos de metodologias ágeis

Ao longo dos anos têm surgido alguns tipos de metodologia ágil. Estas variantes distinguem-se pela importância que dão aos princípios e valores descritos no manifesto de desenvolvimento ágil de *software*. Scrum, XP (Extreme Programming), FDD (Feature Driven Development), ASD (Adaptative Software Development), Crystal e DSDM (Dynamic System Development Method), são algumas aplicações ágeis que ganharam popularidade e onde existiu um maior número de estudos e análises desenvolvidas (Conboy, 2009; Dingsøyr et al., 2012; Hamed & Abushama, 2013; Moniruzzaman & Hossain, 2013). Das diferentes metodologias apresentadas anteriormente apurou-se que as mais utilizadas são Scrum e XP (Abrahamsson, Salo, Ronkainen, & Warsta, 2002).

Scrum é uma metodologia de desenvolvimento que entrega produtos com valor acrescentado aos clientes e que lida com problemas e situações complexas ("The Scrum Guide - Scrum Alliance," 2014). Utiliza uma abordagem iterativa e incremental para desenvolver os seus produtos servindo-se de equipas autónomas e multifuncionais. O seu ciclo de vida passa por *Sprints* que não devem ultrapassar as quarto semanas, e por reuniões diárias de 15 minutos ("The Scrum Guide - Scrum Alliance," 2014). Segundo Rising e Janoff pode ser utilizada em pequenas equipas ou escalar para organizações inteiras (2000). Scrum trabalha tendo em conta que o desenvolvimento de *software* pode ser imprevisível e que os requisitos podem mudar em qualquer altura do projeto (Hamed & Abushama, 2013). Relativamente às entregas, o intervalo entre as mesmas não deve ultrapassar os 30 dias.

Feature Driven Development (FDD) é um método ágil e adaptativo cujo objetivo é afetar apenas dois passos do ciclo de vida do desenvolvimento de *software*, *design* e construção. Existe uma preocupação acrescida relativamente à qualidade dos processos e às entregas frequentes. Segundo Abrahamsson, FDD é adequado para sistemas críticos ao contrário das outras metodologias (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). Foca-se na produção de resultados e a entrega dos mesmos de duas em duas semanas de maneira a diminuir o risco do projeto. A sua maneira de trabalhar é criando pequenos blocos de funcionalidades chamadas *features* que sejam consideradas relevantes pelo cliente/negócio (Abrahamsson et al., 2002).

Dynamic System Software Development (DSDM) é uma metodologia que suporta desenvolvimento de *software* rápido, colaborativo e iterativo na criação de produtos de qualidade. A metodologia trabalha definindo à *priori* a quantidade de recursos e tempo que existe para o desenvolvimento das funcionalidades, sendo que depois os objetivos e

o número de funcionalidades a serem entregues são ajustados (Abrahamsson et al., 2002; Madadipouy, 2015). Para além disto a metodologia segue a filosofia 80/20, isto é, 80% do sistema é entregue em 20% do tempo do projeto (Pressman, 2005).

Adaptative Software Development (ASD) foi uma metodologia proposta para resolver problemas complexos no desenvolvimento de sistemas. O objetivo é guiar o projeto no sentido de evitar falhas mas sempre com a preocupação para a fomentação da criatividade e flexibilidade do processo (Abrahamsson et al., 2002). A adaptação é significativamente mais importante do que a otimização (Highsmith, 2013).

Crystal é constituída por um conjunto de metodologias que se destinam a diferentes tipos de projetos. A criação da metodologia surgiu da tentativa de identificar a relação entre a qualidade do projeto e a equipa de desenvolvimento, e de que a qualidade da equipa influencia a qualidade do produto (Cockburn, 2002). É por essa razão que Crystal permite que seja a equipa a escolher os métodos que mais se adequam ao contexto do projeto (Pressman, 2005). A decisão deve ser tomada consoante o tamanho e risco do projeto (Abrahamsson et al., 2002). O processo é efetuado em quatro fases: identificação do estado, revisão, monitorização e paralelismo (Strode, 2005).

Extreme Programming (XP) é uma metodologia adequada a projetos com interpretação dúbia e com requisitos que mudam com frequência (Beck, 1999). A equipa de desenvolvimento tem grande poder de decisão para responder às mudanças por parte do cliente, mesmo em fases avançadas do projeto. A metodologia lida com mudanças custosas através de ciclos de vida iterativos e com pequenas entregas. Estas entregas são compostas por um conjunto de funcionalidades e são dadas como fechadas após a aceitação do cliente. As mesmas devem ser colocadas em ambiente de produção no prazo de duas semanas (Paulk, 2001). Nesta metodologia, tanto os gestores como os clientes e programadores fazem parte de uma equipa que se dedica à entrega de *software* de qualidade. Dos processos de desenvolvimento desta metodologia destacam-se: *User stories*, programação a pares, foco nos testes e integração contínua (Meso, 2006).

Segundo, (Wang, Conboy, & Cawley, 2012), apesar das metodologias diferirem nas especificações técnicas, têm muito em comum, incluindo ciclos de vida curtos e iterativos, *feedback* rápido e frequente dos clientes, e aprendizagem constante.

A tabela exposta abaixo (Tabela 3) apresenta as similaridades e diferenças entre os métodos ágeis apresentados acima. No Anexo A encontra-se uma análise detalhada das características de cada metodologia ágil.

Tabela 3 – Diferenças entre as metodologias ágeis

Critério	**XP**	***Scrum***	**FDD**	**ASD**	**DSDM**	***Crystal***
Tamanho do projeto	Pequenos e médios	Pequenos, médios e grandes	Pequenos, médios e grandes	Grandes e complexos	Pequenos e grandes	Pequenos e médios
Tamanho da equipa	<10	<10 e várias equipas	Sem limite	N/A	Máximo 6 e várias equipas	Máximo 6 equipas singulares; Equipas múltiplas no máximo entre 40 a 80 pessoas
Desenvolvimento	Iterativo e rápido	Iterativo e rápido	Iterativo	Iterativo e rápido	Iterativo, rápido e cooperativo	Iterativo e rápido
Cultura	Colaborativa e cooperativa	N/A	N/A	N/A	Colaborativa e cooperativa	N/A
Estilo de programação	Orientada aos objetos	Orientada aos objetos	Orientada aos objetos	Orientada aos objetos /Orientada às componentes	Orientada aos objetos /Orientada às componentes	Orientada aos objetos

2.3. Paradigma *Lean*

Na sequência da revisão literária, e tendo em conta a quantidade de informação encontrada que relaciona este tema, Paradigma *Lean*, com as metodologias ágeis, considerou-se essencial abordar o conceito. Nesse sentido, nesta subsecção ir-se-á analisar qual a relação/contributo do paradigma *Lean* com as metodologias ágeis.

O conceito *Lean* surgiu nos anos oitenta (80) com a revolução na redução de desperdícios na produção de carros pela Toyota (Ohno, 1988). Ao longo dos anos, a metodologia foi ramificando para outras indústrias e para outras áreas de negócio.

Desenvolvimento de *software Lean* consiste no desenvolvimento de um produto com o foco na criação de valor para os clientes, eliminação de desperdício, otimização dos processos, delegação de poder pelas pessoas e aperfeiçoamento contínuo (Ebert, Abrahamsson, & Oza, 2012).

Nos últimos anos, a comunidade das metodologias ágeis tem-se voltado para as técnicas de desenvolvimento de *software Lean* em paralelo com métodos ágeis como *XP* e *Scrum*. O crescimento do número de conferências sobre este tema demonstra também o aumento no seu interesse e da sua propagação (Wang et al., 2012).

Apesar do aumento no interesse pelo paradigma *Lean* no desenvolvimento de *software*, as opiniões relativas ao relacionamento entre o mesmo e os métodos ágeis são díspares.

Jalali e Wohlin (2010), não fazem qualquer distinção entre o paradigma *Lean* e as metodologias ágeis (Jalali & Wohlin, 2010). Já para Dall'Agnol, os conceitos Lean e Ágil destinam-se a diferentes ocasiões (2003). O mesmo afirma que a metodologia XP contém uma série de práticas que são desenhadas para a utilização por parte dos *developers*, mais propriamente para aliviar a tensão entre estes e os clientes. Pelo contrário, o conceito *Lean* aplica-se numa perspetiva superior de gestão, com o objetivo de otimizar as atividades em toda a organização. Barton (2009) identifica que muitas organizações quando aplicam as práticas de *Scrum* no desenvolvimento de *software*, consideram que todo o trabalho envolvente é uma implementação *Lean*.

Mary e Tom (2006) e Poppendieck e Poppendieck (2003) acreditam que o desenvolvimento *Lean* complementa a fundação teórica do desenvolvimento ágil por aplicar princípios *Lean* aceites pela comunidade. Os mesmos sugerem que os princípios *Lean* são guias para o desenvolvimento e adaptação das práticas ágeis. (Perera & Fernando, 2007), em congruência, refere que aplicar técnicas *Lean* ajuda a estabilizar o desenvolvimento ágil especialmente nas fases mais avançadas do projeto.

Apesar de tanto o paradigma Ágil como o *Lean* apresentarem muitas semelhanças, nomeadamente em termos de simplicidade e qualidade, existe uma grande diferença (Conboy & Fitzgerald, 2004). A premissa pela qual o *Lean* se rege é "eliminar todo o desperdício" (Conboy & Fitzgerald, 2004). Já a metodologia Ágil necessita que o desperdício seja eliminado, mas apenas se a capacidade de resposta à mudança não seja afetada (Conboy & Fitzgerald, 2004).

Alguns artigos durante a pesquisa referem a técnica Kanban. A mesma não foi incluída na revisão pois considerou-se que a mesma é baseada nos princípios *Lean,* no sentido em que pretende remover o desperdício no processo de produção valorizando a visualização do *work flow*, a limitação do trabalho em progresso, e a medição do tempo para finalizar as tarefas (Ikonen, Kettunen, Oza, & Abrahamsson, 2010).

Ao longo da pesquisa, verificou-se que não existe consenso entre os autores relativamente às diferenças entre o paradigma *Lean* e Ágil, havendo quem não faça qualquer distinção entre as duas. Esta semelhança, segundo Wang (2012), deve-se também ao facto de algumas das práticas de desenvolvimento *Lean* já terem sido estabelecidas nos métodos ágeis (exemplos de práticas: melhoria contínua para estabelecer um fluxo mais suave, tornar o estado do projeto visível, medir e gerir, deteção de erros, etc) (Wang et al., 2012).

Relativamente aos estudos efetuados, como os dois conceitos são muito abrangentes e as suas premissas não estão totalmente definidas, as interpretações criadas e as definições estabelecidas variam de autor para autor.

2.4. Implementação das metodologias ágeis no sector público

Apesar do aumento de implementação de metodologias ágeis nos últimos anos, a adoção dos mesmos no sector público tem sido mais lenta. Este facto reflete-se também na literatura académica, já que são escassos os estudos existentes relativos à adoção das metodologias ágeis no sector público (Kaczorowska, 2015; Karaj & Little, 2013; Powner, 2012). Atualmente, o uso de desenvolvimento ágil está a ganhar notoriedade nas instituições públicas, no entanto, tem sido escassa a atenção dada aos estudos empíricos que identificam e descrevem os desafios relativos à implementação e execução das metodologias ágeis no contexto do sector público (Nuottila, Aaltonen, & Kujala, 2016).

Da mesma maneira, Conboy fala sobre os desafios da adoção das práticas ágeis e solicita mais investigação nos efeitos da implementação das metodologias ágeis (Conboy & Wang, 2007). Além disso, Conforto et al. (2014) considera importante a existência de mais pesquisa na gestão de projeto ágil e no uso das práticas ágeis no desenvolvimento de *software* noutras indústrias.

Thomas, R, Niederman F (2017) referem também a importância de mais pesquisa na implementação ágil em diferentes contextos.

Asnawi, Gravell, & Wills (2011) identificaram que para algumas organizações o uso de metodologias ágeis tornava-se mais difícil quando a trabalhar para entidades públicas, pois as mesmas não estavam familiarizadas com as práticas ágeis. Asnawi et al. (2011) também observou que a taxa de rotatividade das equipas era alta em organizações governamentais, o que tornava desafiante a utilização eficiente das metodologias ágeis.

Organizações públicas têm algumas características que tornam a aquisição de *software* mais desafiante comparativamente às organizações privadas. Para além de existir uma regulação legal na aquisição de projetos e processos relativos (Edquist, Hommen, & Tsipouri, 2000), os sistemas tecnológicos deste tipo de organizações são, por norma, muito grandes e complexos (Brown, 2001). A inovação e a velocidade de desenvolvimento também são, normalmente, mais lentas do que no sector privado (Brown, 2001). Adicionalmente, a falta de gestão adequada tem sido falada como um fator preponderante que causa dificuldades nos projetos do sector público (Brown, 2001). Parker e Bradley (2000) referem que a cultura organizacional no sector público é, frequentemente muito hierarquizada, com regras e políticas pouco flexíveis e com documentação muito formal e orientada à comunicação. O uso da abordagem ágil na gestão de sistemas de Tecnologias de Informação no sector público é também exposto a dificuldades causadas pelos princípios de seleção e cooperação do fornecedor e pela

capacidade do mesmo conseguir implementar o projeto (Kaczorowska, 2015). Kaczorowska (2015) também menciona que os projetos públicos no começo do projeto exigem uma descrição detalhada da solução, ficando difícil introduzir mudanças subsequentes.

O estudo efetuado por Wisitpongphan e Khampachua (Wisitpongphan & Khampachua, 2016), identifica dois desafios na implementação das metodologias ágeis. O primeiro é relativo à participação dos utilizadores finais no desenvolvimento do projeto. Era apenas possível contactar por email ou contacto telefónico, sendo que o tempo de resposta era elevado. O segundo problema foram os requisitos adicionais integrados após o *kick-off* do projeto. Os projetos tinham um orçamento fixo que foi determinado na fase de ante-projeto e estas novas *features* criam custos extra para as organizações.

Já Jouko Nuottila e Kirsi Aaltonen (2016) identificaram sete categorias de desafios na implementação de metodologias ágeis no sector público (Nuottila et al., 2016):

- Documentação – *Software* funcional **em detrimento** de **documentação clara/responsiva** por vezes é mal **interpretado pela** inexistência de documentação;
- Educação, experiência e dedicação – Tanto em termos organizacionais como individuais **é necessário existir** envolvimento e dedicação;
- A comunicação e envolvimento dos *stakeholders* – É possível que o sistema a ser desenvolvido tenha dependências com outros. Nesse sentido, é importante identificar os *stakeholders* o mais cedo possível e comunicar todas as decisões relevantes bem como as mudanças ao longo do projeto;
- Papéis dos intervenientes no projeto – A mudança da metodologia de desenvolvimento **conduz a mudanças** nos **papéis desempenhados**, não só **pelos membros das equipas** como dos intervenientes do negócio. A mudança das responsabilidades e tarefas dos colaboradores por norma não é bem recebida devido à resistência que existe na saída da zona de conforto.
- Localização da equipa de desenvolvimento – No estudo do autor também foram identificadas equipas que trabalhavam à distância dificultando a coordenação, comunicação e cooperação entre a equipa e os restantes *stakeholders;*
- Legislação – O estudo aponta para alguns confrontos entre a legislação e os princípios da metodologia implementada. A comunicação transparente com os *stakeholders*, por exemplo, por vezes não podia ser realizada. Outro problema identificado é relacionado com as datas de entrega, já que muitas vezes as datas

impostas legalmente eram divulgadas tendo em conta o método cascata. Ao contrário desse, as práticas ágeis geralmente usam integração contínua que pode ser posta em causa com este tipo de constrangimentos;

- Complexidade da arquitetura dos sistemas de *software* e de integração – Segundo o estudo existiram algumas dificuldades na integração de sistemas e, originalmente, os métodos ágeis foram criados para sistemas pequenos e isolados.

Apesar dos desafios apresentados, Karaj e Little (2013), no seu artigo, reportam um caso de sucesso na adoção das práticas ágeis no sector público.

Um estudo efetuado pela Universidade de Sevilha (Torrecilla-Salinas, Sedeño, Escalona, & Mejías, 2013) apresenta os resultados da aplicação das metodologias ágeis (*Scrum*, neste caso) numa organização pública. Neste estudo foi utilizada uma técnica de planeamento e estimação para os projetos em questão e verificou-se que com a aplicação de *Scrum* a maior parte dos resultados obtidos foram ao encontro das estimações iniciais.

Um estudo efetuado por Roses, Windmöller e (2016) que tinha como objetivo avaliar as condições na adoção das metodologias ágeis no sector público, concluiu que quando confrontados com a mesma, existe preferência na sua utilização.

Apesar da resistência à mudança e dos desafios reportados nos estudos apresentados anteriormente, é possível verificar que existem casos de sucesso na implementação das metodologias ágeis no sector público.

2.5. Estratégias de gestão da mudança

Tal como foi apresentado no segundo capítulo, subsecção 2.1 e 2.4, a resistência à mudança é um dos desafios na implementação das metodologias ágeis. Na subseção 2.4, é ainda referido que as entidades públicas apresentam-se, por norma, como mais resistentes à mudança. Nesse sentido, achou-se relevante dedicar uma subsecção a esse problema. O presente sub-capítulo tem como objetivo explorar as causas desta problemática e as estratégias existentes para a sua mitigação.

Para Conboy, Coyle, Wang e Pikkarainen (2011) e Lalsing, Kishnah, e Pudaruth (2012) a consciencialização dos colaboradores relativamente aos métodos ágeis pode ser um fator chave para que percebam quais os benefícios da sua aplicação. No entanto, segundo Fontana, Fontana, da Rosa Garbuio, Reinehr e Malucelli (2014), tem havido falta de preocupação com este tema na implementação das práticas.

A adaptação de novos métodos de desenvolvimento de *software* normalmente não é tarefa fácil, já que, é necessário mudar as atitudes e valores dos membros da organização (Chan & Thong, 2009).

Roberts, Gibson, Fields, e Rainer (1998) referem também que a aceitação de novos processos de desenvolvimento representa uma mudança mais significativa do que a adoção de novas tecnologias e ferramentas.

Na revisão literária relativa a este tema foram encontradas algumas teorias quanto aos fatores que influenciam a aceitação dos métodos de desenvolvimento de *software*.

Riemenschneider, Hardgrave e Davis (2002) criaram uma *framework* de aceitação de ferramentas de Tecnologias de Informação que são constituídas por um conjunto de mecanismos que permitem a medição das características, como a utilidade e facilidade de utilização, das metodologias de desenvolvimento de *software*.

Sultan e Chan (2000) referem que a aceitação das novas metodologias de desenvolvimento centra-se nas características individuais e culturais das organizações. Ihlsoon Cho (2002), no seu estudo, enfatiza a importância dos fatores organizacionais na aceitação dos novos processos de desenvolvimento.

A tabela que se apresenta abaixo (Tabela 4) apresenta as causas da resistência à mudança publicadas no artigo (Chan & Thong, 2009). Para complementar estes dados foram integrados mais dois artigos com data posterior a 2009.

Tabela 4 – Causas da resistência à mudança organizacional

Causas de resistência	**Definição**	**Referências**
Cultura organizacional	A cultura organizacional bem como os seus objetivos têm de estar alinhados com a adoção das novas metodologias.	(Gandomani, Zulzalil, Azim, & Ghani, 2014) (Sulaiman, Mahrin, & Yusoff, 2016)
Utilidade percebida	Refere-se às implicações que um colaborador espera ter na adoção das metodologias ágeis .	(Chan & Thong, 2009)
Facilidade de utilização percebida	Refere-se à crença que um colaborador tem na simplicidade do esforço das funções que desenvolve.	(Chan & Thong, 2009)
Compatibilidade percebida	Refere-se à compatibilidade que existe entre a adoção da nova metodologia e as os antigos processos de desenvolvimento.	(Chan & Thong, 2009)
Acessibilidade dos resultados	Refere-se à acessibilidade que existe na demonstração e tangibilidade das vantagens das metodologias de desenvolvimento.	(Chan & Thong, 2009)
Maturidade percebida	Refere-se à incerteza e inexperiência percebida da metodologia ágil.	(Chan & Thong, 2009)
Falta de conhecimento	Refere-se à falta de conhecimento das vantagens na implementação de metodologias ágeis.	(Gandomani et al., 2014) (Sulaiman et al., 2016)

A tabela 5 apresenta as estratégias que mitigam os desafios identificados na tabela anterior.

Tabela 5 – Estratégias de mitigação na resistência à mudança organizacional

Estratégias de mitigação	**Definição**	**Referências**
Treino	Procedimentos que permitem a aprendizagem dos métodos por parte dos colaboradores.	(Chan & Thong, 2009) (Sulaiman et al., 2016)
Suporte externo	Suporte externo refere-se ao uso externo de informação relativamente aos benefícios das metodologias ágeis.	(Chan & Thong, 2009)
Implicações na carreira	Refere-se às consequências na carreira aquando da utilização das metodologias.	(Gandomani et al., 2014)
Suporte dos gestores	Suporte dos gestores é fundamental para o acompanhamento e motivação da equipa ao longo do desenvolvimento.	(Sulaiman et al., 2016) (Gandomani et al., 2014)
Voluntariado	Voluntários que defendam a utilização de novas metodologias são catalisadores na adopção das mesmas nos restantes colaboradores.	(Gandomani et al., 2014)

Capítulo 3 – Contexto Organizacional

O seguinte capítulo pretende descrever o contexto organizacional do IGFEJ, focando-se na gestão de projetos de desenvolvimento de software e nos problemas inerentes à mesma. A informação apresentada foi baseada nos documentos institucionais partilhados e nas entrevistas e interações experienciadas.

Relativamente à metodologia de investigação este capítulo insere-se na fase de 'Identificação dos Problemas' da *Design Science Research.*

Tal como já foi referido anteriormente o contexto organizacional onde esta dissertação se insere é o IGFEJ, instituição pública Portuguesa, autónoma administrativa e financeiramente que prossegue atribuições do Ministério da Justiça, sob a sua superintendência e tutela.

Assume um carácter central no Ministério da Justiça, prestando serviços para o bom funcionamento do sistema judiciário. O IGFEJ atua em várias vertentes do Ministério da Justiça, nomeadamente orçamental e financeira, patrimonial e obras, infraestruturas tecnológicas e sistemas de informação.

Tem também as seguintes responsabilidades: Gestão financeira e orçamental do Ministério da Justiça, monitorizar as custas processuais e o apoio judiciário, assegurar a gestão e manutenção do património do Ministério da Justiça e dinamizar e desenvolver sistemas de informação e o reforço das infraestruturas tecnológicas da Justiça.

Relativamente à estrutura, o IGFEJ está dividido em vários departamentos apresentados na Figura 1:

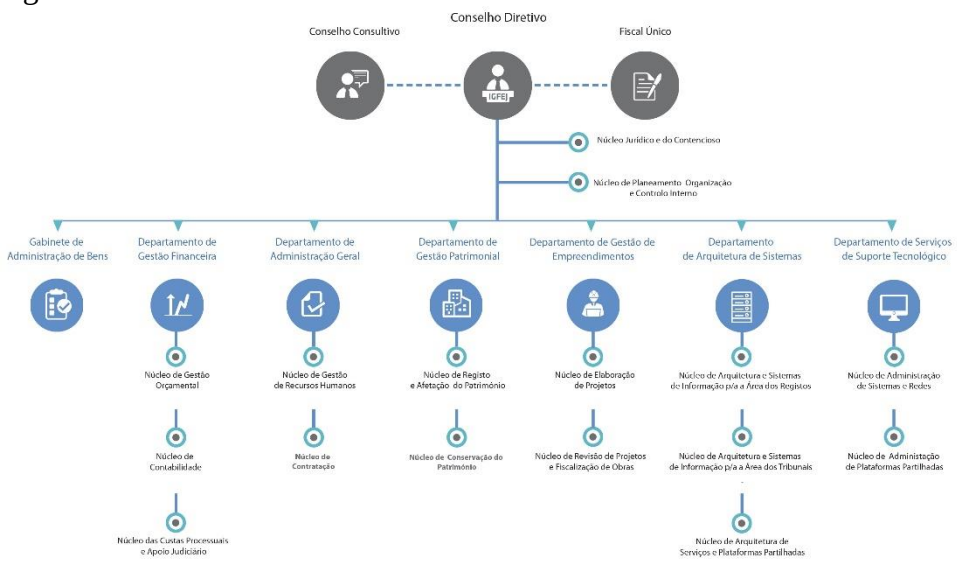


Figura 1- Organograma do IGFEJ

Os departamentos relacionados com os sistemas de informação são: 1) Departamento de Arquitetura de Sistemas, que é responsável pelo desenvolvimento e manutenção dos sistemas de informação da justiça; 2) Departamento de Serviços de Suporte Tecnológico, responsável por toda a infraestrutura de suporte aos S.I.
Estes departamentos trabalham em conjunto para atingir os seguintes objetivos:

- Assegurar a execução e manutenção dos recursos tecnológicos e dos sistemas de informação da Justiça, garantindo a sua gestão e administração, em articulação com os demais serviços e organismos do Ministério da Justiça, e o apoio informático aos respetivos utilizadores.

- Assegurar a adequação dos sistemas de informação às necessidades de gestão e operacionalidade dos órgãos, serviços e organismos da área da Justiça.

- Elaborar propostas de articulação com o plano estratégico dos sistemas de informação da área da Justiça, tendo em atenção a evolução tecnológica e as necessidades globais de formação.

- Elaborar, desenvolver e coordenar propostas de projetos de investimento, em matéria de informática e comunicações dos serviços e organismos do Ministério da Justiça.

- Promover soluções de gestão de informação estruturada e não estruturada na área da Justiça, designadamente de acesso geral, nas áreas jurídica e documental.

Relativamente ao desenvolvimento de *software* no IGFEJ, não foi identificada nem é assumida pela organização nenhuma metodologia de desenvolvimento, pelo que cada colaborador tem a liberdade de poder adotar os métodos que considera mais adequados. Os projetos por norma são apresentados através das novas legislações e nesse sentido, os prazos têm obrigatoriamente de ser cumpridos, sendo que o seu incumprimento tem consequências legais. Outra característica destes projetos relaciona-se com a volatilidade dos requisitos, uma vez que estes podem ser alterados como consequência da publicação de novas Portarias Regulamentares durante o decorrer do projeto.

Com este cenário, foram distinguidos dois tipos de projetos.

O primeiro tipo, são projetos desenvolvidos internamente na organização e, por norma, associados aos Sistema de Informação do IGFEJ, isto é, projetos que estão relacionados com alterações nos Sistema de Informação atualmente existentes.

O segundo tipo, são projetos desenvolvidos externamente (chave na mão), totalmente alocados a um prestador de serviço. Por norma, criam outros sistemas de informação, ou novas funcionalidades, que interagem com os existentes.

Em ambos os tipos de projetos foram identificados problemas na gestão de projetos de desenvolvimento de *software* que levam ao incumprimento dos três elementos chave da gestão de projeto mencionados no PMBOK (Larson and Gray 2015): Atrasos na entrega integral do projeto, entrega de funcionalidades defeituosas e derrapagens no custo.

No primeiro caso (projetos internos), estes problemas evidenciam-se na ausência de práticas associadas à gestão de projeto. Isto é, elaboração de:

1. Documentação - A documentação, em cerca de 90% dos projetos não é elaborada e nos restantes casos a mesma é construída com base em templates criados pelo gestor de cada projeto, não havendo uniformização da informação;
2. Planeamento do projeto - O planeamento do projeto é, tal como documentação, na maior parte das vezes deixado de parte. O foco é o desenvolvimento e na entrega atempada das funcionalidades;
3. Levantamento dos riscos - O levantamento dos riscos não é, em caso algum, efetuado. Como não existe a possibilidade de cancelar o projeto ou alterar a data de entrega do mesmo não existe preocupação em efetuar o levantamento dos riscos;
4. Definição do âmbito - O âmbito do projeto é, por norma, identificado na legislação que dita a criação do projeto. No entanto, o âmbito não é transposto de maneira clara para a equipa. A mesma para se voltar a enquadrar no tema terá de rever a legislação.
5. Definição do papel dos *stakeholders* - A identificação dos *stakeholders* é efetuada informalmente (não existe nenhum documento ou ferramenta que contenha esta informação). As consequências que se verificaram no IGFEJ por esta questão não ser formalizada foram: 1ª Desresponsabilização dos colaboradores nas falhas ocorridas; 2ª Sobreposição de tarefas. Isto é, dois colaboradores a efetuarem a mesma tarefa ou a tomarem decisões diferentes relativas ao mesmo assunto;
6. Formalização dos requisitos - Muitas vezes os próprios requisitos não eram documentados. A informação era transmitida via email e os desenvolvimentos eram efetuados a partir desse meio de comunicação. Este ponto teve impacto na

elaboração dos projetos, na medida em que requisitos eram perdidos/esquecidos na *thread* de emails.

Estes são pontos fundamentais para a correta gestão de projeto segundo o PMBOK (Larson and Gray 2015). No caso do IGFEJ cada gestor tem autonomia para escolher de que maneira pretende gerir, e como a prioridade passa sempre pela entrega de *software*, estes pontos, por norma são escamoteados. A falta de uniformização da informação leva também a que a comunicação e troca de dados entre departamentos da empresa não seja fluída que, por conseguinte, leva a atrasos no desenvolvimento.

No segundo tipo de projetos (chave na mão), verifica-se a ausência de gestão interna nos projetos. No decorrer do projeto, por norma, existem apenas duas interações com a equipa de desenvolvimento: uma inicial onde é apresentado o projeto e uma segunda correspondente à entrega do mesmo. Consequentemente o controlo e monitorização do trabalho efetuado é quase nulo. A equipa externa tem total autonomia no desenvolvimento e não existe gestão interna de projeto, sendo que, a única obrigação é a entrega final do produto. Esta falta de sinergias entre o IGFEJ e o prestador de serviços leva a que os problemas inicialmente indicados (falhas no prazo de entrega, funcionalidades defeituosas e/ou mal interpretadas e custos acrescidos) ocorram sem que a organização tenha conhecimento até à data de entrega do projeto.

Capítulo 4 – Definição do modelo a implementar

O modelo tem como objetivo servir como guia para a implementação de uma metodologia no desenvolvimento de *software* do IGFEJ baseada nas metodologias ágeis estudadas no capítulo anterior.

O modelo, apresentado na Figura 2, é composto por duas fases. A primeira fase refere-se às estratégias de gestão da mudança face à possível resistência à mesma, de modo a que a adoção da metodologia seja eficiente e passível de causar o menor tipo de entropia. A segunda passa pela definição da metodologia ágil de desenvolvimento de software a adotar num projeto desenvolvido no IGFEJ.

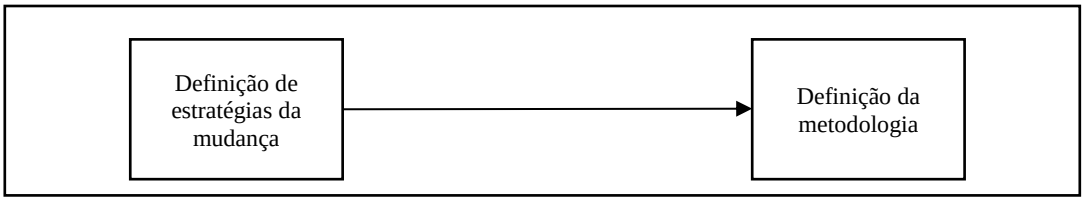


Figura 2 - Esquema do modelo para a implementação de uma metodologia ágil no sector público

Relativamente à metodologia de investigação da *Design Science Research*, este capítulo enquadra-se nas fases de ‘Sugestão de uma Solução’ e ‘Desenvolvimento das ações necessárias para ir de encontro à solução’.

4.1. Definição das estratégias de gestão da mudança

Tal como foi descrito no capítulo 1 a resistência à mudança é um dos fatores de insucesso aquando da implementação de metodologias ágeis no sector público. Estas apresentam uma metodologia de trabalho diferente das tradicionais. São exemplo de práticas associadas aos métodos ágeis: entregáveis mais frequentes, maior comunicação entre *stakeholders* e maior liberdade para alteração de requisitos durante o projeto. Nesse sentido, irá fazer parte do modelo criado um conjunto de estratégias de gestão da mudança. Com base no levantamento efetuado na subseção 2.5, estratégias de gestão da mudança, foram definidas 3 etapas: 1ª Formação, 2ª Exercício Prático e 3ª Avaliação.

A Formação é composta por uma apresentação onde são lecionados os conceitos de metodologia ágil, suas vantagens e desvantagens e comparação das mesmas com as tradicionais. Adicionalmente, e com maior ênfase, é apresentada a metodologia criada e proposta no âmbito da presente dissertação.

Após serem apresentados os conceitos anteriores, é efetuado um exercício prático que consiste na aplicação num projeto real da metodologia ágil lecionada na primeira fase. O objetivo desta etapa é que os colaboradores tenham oportunidade de, em ambiente experimental, interagirem e aplicarem as práticas ágeis sem qualquer tipo de compromisso.

Por último deve ser efetuada uma avaliação qualitativa à metodologia exercitada na fase anterior, no sentido de recolher *feedback* construtivo e assim melhorar o método ágil a implementar no contexto do projeto do IGFEJ. Para além disso, o contributo dos colaboradores, através da avaliação, faz com que a sua relação com as novas práticas seja maior e, por conseguinte, que a gestão da mudança seja agilizada.

Estas medidas foram introduzidas na organização através de um *workshop* e têm como o objetivo a consciencialização dos colaboradores para os benefícios da adoção das metodologias ágeis no seu quotidiano, mitigando assim, a resistência à adoção das mesmas e aumentando a sua recetividade. Desta maneira procura-se colmatar cerca de 70% das causas da resistência à mudança apresentadas da subsecção 2.5 (Utilidade percebida, Facilidade de utilização percebida, Compatibilidade percebida, Maturidade percebida e Falta de conhecimento), bem como, os seguintes desafios na implementação das metodologias ágeis apresentados na tabela 1 da subsecção 2.1: 1ª Ausência de um projeto piloto; 2ª Falta de capacidade para alterar a cultura organizacional; 3ª Falta de experiência com o novo método; 4ª Treino disfuncional e inadequado; 5ª Dificuldade em

convencer a equipa de topo a investir num novo método; 6ª Existência de mentes mais fechadas para a entrada de novas práticas.

4.2. Definição da Metodologia proposta

O processo de criação da metodologia a implementar no âmbito de um projeto no IGFEJ foi iterativo pelo que passou por várias fases de desenvolvimento, tal como é demonstrado na Figura 4. Pode-se verificar que a metodologia teve uma primeira versão constituída com base na análise da realidade atual do IGFEJ relativamente aos processos de desenvolvimento e gestão de projetos, em entrevistas aos colaboradores do IGFEJ e na informação recolhida na revisão literária elaborada no segundo capítulo da dissertação.

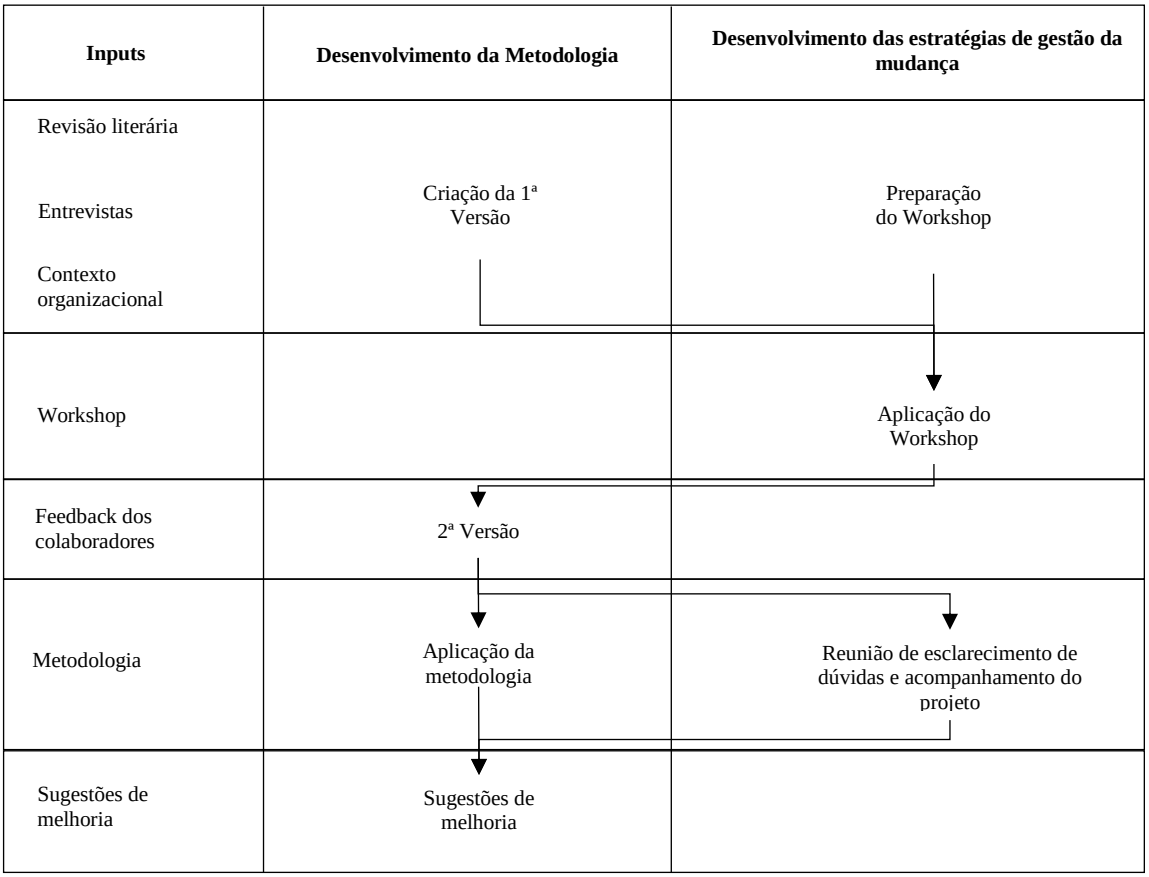


Figura 3 – Evolução do desenvolvimento da metodologia

Posteriormente ao processo de proposta da metodologia, foi efetuado o *workshop* a colaboradores do IGFEJ (elemento que constitui as estratégias da gestão da mudança) onde foi recolhido *feedback* quantitativo e qualitativo relativo à aceitação da metodologia. Com este *input* foi criada uma nova versão da metodologia, definida de seguida.

Como resultado da revisão literária apresentada anteriormente, onde se identificaram e descreveram as metodologias de desenvolvimento ágil de *software*, verificou-se que as mais utilizadas são XP e Scrum (Cockburn, 2002). Para além deste facto, na análise de estudos semelhantes ao abordado na presente dissertação, isto é, implementações de

metodologias ágeis no sector público, identificou-se uma clara preferência pelo método Scrum (todos os estudos empíricos aplicaram Scrum - Roses, Windmöller, & Carmo, 2016; Torrecilla-Salinas, Sedeño, Escalona, & Mejías, 2013; Karaj & Little, 2013; Nuottila et al., 2016; Wisitpongphan & Khampachua, 2016).

Relativamente ao desenvolvimento de *software* no IGFEJ existem problemas no controlo e gestão que levam a atrasos nas entregas, a funcionalidades pouco robustas, custos acrescidos e a conflitos interpessoais. Tendo em conta que o Scrum é uma metodologia incremental, iterativa e flexível, faz com que se adeque na mitigação deste tipo de problemas. A sua capacidade de alteração de requisitos a qualquer altura do projeto faz com que, quando a publicação de legislação obrigue a novas implementações, as mesmas possam ser integradas durante os desenvolvimentos. Para além disso, esta metodologia permite revisões e replaneamentos do projeto no fim de cada sprint, mitigando desvios do âmbito, conflitos e funcionalidades defeituosas. As restantes metodologias apresentam algumas limitações na correção dos problemas identificados. Crystal entrega toda a liberdade à equipa para definir a metodologia com que pretende trabalhar, o que resulta mais uma vez, numa gestão de projeto não uniformizada. FDD trabalha com base em *features*, algo que não se demonstra vantajoso ao contexto do IGFEJ uma vez que não se foca nas práticas que permitem suprimir os problemas existentes na organização identificados no capítulo 3. DSDM não se considera viável neste âmbito, não abordando o tema da gestão e monitorização do projeto. Por fim, ASD apresenta-se como mais adequada para projetos complexos escamoteando os custos e recursos para elaborar os mesmos, sendo que o IGFEJ necessita de controlar o esforço monetário associado a cada proposta.

Considerou-se que deveria ser apresentada à empresa uma metodologia ágil com conteúdos mais diretivos e com menos espaço para interpretações subjetivas. Nesse sentido propõe-se uma adaptação de uma metodologia ágil e não a aplicação literal de uma já existente. Esta preocupação tem em consideração o facto de a abordagem ágil ser pela primeira vez introduzida num projeto do IGFEJ. Os conceitos (embora conhecidos pelos colaboradores) ainda não foram experimentados, e é necessária uma orientação clara e eficaz na aplicação dos mesmos. A flexibilidade de utilização da metodologia fará mais sentido em organizações cujas práticas ágeis estejam já enraizadas.

Neste sentido, a metodologia será baseada nos métodos de Scrum e nas áreas de conhecimento do PMBOK e caracteriza-se pela definição das seguintes dimensões. 1)

Documentação do projeto, 2) Responsabilidades das tarefas, 3) Fluxo de informação e 4) Práticas de desenvolvimentos de software.

4.2.1. Documentação do projeto

O funcionamento do Scrum, em conformidade com as restantes metodologias ágeis, defende a redução da documentação comparativamente à elaborada pelas metodologias tradicionais. O objetivo é produzir apenas a documentação necessária ao desenvolvimento do projeto privilegiando a evolução técnica da solução.

Nesse sentido é proposta a elaboração de um conjunto de documentos complementados pela apresentação dos respectivos *templates* nos apêndices A,B,C,D e E da presente dissertação:

- Lista de requisitos – Este documento deve incluir todas as funcionalidades, *bugs*, defeitos, *updates*, documentação e melhorias que o produto vai sofrer. Este documento deve permitir que os *stakeholders* tenham acesso ao progresso dos desenvolvimentos do projeto. O *template* proposto para o documento encontra-se no Apêndice A.
- Relatório de progresso – No fim de cada *sprint* deve ser preenchido o relatório de progresso que tenciona centralizar a informação relativa ao estado em que o projeto se encontra após o *sprint*. O objetivo é perceber se o *sprint* foi concluído com sucesso ou se ficou trabalho por elaborar. O *template* proposto para o documento encontra-se no Apêndice C.
- Plano de projeto – O plano de projeto contém toda a informação do projeto. Todas as alterações efetuadas aos requisitos e aos *sprints* devem ser registadas neste documento. Âmbito, *stakeholders*, pressupostos e dependências é informação que também deve ser incluída, bem como ciclo de vida do projeto, entregáveis e *milestones*. O *template* proposoto para o documento encontra-se no Apêndice E.

Para além destes documentos, dependendo do tipo de projeto deve ser incluída documentação sobre:

- Guião de utilização – Dependendo de cada projeto, se se considerar pertinente, deve-se elaborar um manual de utilização do sistema com o objetivo de auxiliar os utilizadores no manuseamento do sistema.

- Documentação técnica – O código desenvolvido deve ser comentado. Este ponto tem especial importância nos algoritmos mais complexos e que possam levantar mais dúvidas na sua compreensão.
- Project Charter – Este documento deve ser desenvolvido pelo cliente que atribui o projeto ao IGFEJ. Contém informações básicas sobre o projeto de modo a agilizar o início do projeto e clarificar algumas componentes do mesmo. O *template* proposto para o documento encontra-se no Apêndice B.
- Requisito de mudança – Este documento contém informação sobre requisitos que se pretende alterar ou acrescentar no projeto. Alterações que sejam significativas faz sentido que sejam requeridas formalmente através do presente documento para que haja a possibilidade de rastrear as mudanças efetuadas nos requisitos. O *template* proposto para o documento encontra-se no Apêndice D.

4.2.2. Responsabilidades e papéis

A importância da definição das responsabilidades e papéis é referenciada na definição das diversas metodologias ágeis. Para além disso, verifica-se no PMBOK uma área de conhecimento dedicada à análise das mesmas. Com base nos dois factos referidos anteriormente, considerou-se importante especificar papel dos colaboradores no projeto.

Pretende-se que todos os *stakeholders* conheçam o seu papel e as suas responsabilidades no projeto. Nesse sentido, ir-se-á apresentar a função de cada interveniente nas fases de desenvolvimento, bem como, no preenchimento da documentação anteriormente enunciada.

De seguida apresenta-se os *stakeholders* com responsabilidades no ciclo de vida do projeto:

Equipa de desenvolvimento (EQ) – Equipa de programadores que desenvolvem as funcionalidades do projeto;

Gestor de projeto interno (GPI) – Gestor do IGFEJ que controla o projeto internamente.

Gestor de projeto externo (GPE) – Gestor externo que controla a equipa de desenvolvimento.

Responsável pelas infraestruturas (RI) – Colaborador que tem autoridade para tomar decisões relativamente às infraestruturas necessárias para a elaboração do projeto.

Direção do IGFEJ (DI) – Deve ser nomeada uma pessoa para que represente o IGFEJ e se responsabilize pelo projeto. Um Vogal ou um diretor de departamento poderão assumir este papel.

Ministério Justiça (MJ) – No sentido lato é o cliente que requisita os serviços ao IGFEJ. No entanto, normalmente, é representado por um grupo de trabalho constituído por elementos do Ministério e por utilizadores finais.

Para cada tipo de projetos (projetos internos ou chave-na-mão) desenvolvidos no IGFEJ elaborou-se uma tabela RACI que apresenta as responsabilidades associadas às fases e documentos. Foi feita esta distinção uma vez que os intervenientes em cada tipo de projeto são diferentes resultando numa mudança nos papéis e responsabilidades dos mesmos. Para o primeiro tipo de projetos (projetos internos) a tabela é a seguinte (Tabela 6):

Tabela 6 - Tabela RACI para projetos internos

RACI	**DI**	**MJ**	**GPI**	**EQ**	**RI**
Fases					
Levantamento de requisitos	I	A	R		
Planeamento dos Sprints	I	I/C	A/R	C	I
Execução do Sprint		C	A	R	R
Revisão do Sprint	I	A	R	R	R
Avaliação do projeto	I	A	R	I	
Replaneamento do Sprint	I	I/C	A/R	C	C
Balanço do projeto	I		A/R	C	C
Documentos					
Lista de requisitos		A	R		
Relatório de progresso	I	I	A	R	
Plano de projeto	I	I/C	A/R	C/I	I
Project charter	I	R/A	R/I/C		

Requisito de mudança		R/A	R/A	C/I	

Nota:

A …….: Imputável *(Accountable)* C …….: Consultado

R …….: Responsável I ……...: Informado

Para os projetos chave-na-mão a tabela RACI é a seguinte (Tabela 7):

Tabela 7 - Tabela RACI para projetos chave-na-mão

RACI	**DI**	**MJ**	**GPI**	**GPE**	**EQ**	**RI**
Fases						
Levantamento de requisitos	I	A	R			
Planeamento dos Sprints	I	I/C	A/C	R	C	C
Execução do Sprint		C	A	R	R	R
Revisão do Sprint	I	A	R	R	R	R/C
Avaliação do projeto	I	A	R			
Replaneamento do Sprint	I	I/C	A	R	C/I	C
Balanço do projeto	I		A/C	R	C	C
Documentos						
Lista de requisitos		A	R			
Relatório de progresso	I	I	A	R	C/I	
Plano de projeto	I	I/C	A	R	I	C
Project charter	I	R/A	R/I/C			
Requisito de mudança		R/A	R/A	C/I		

Nota:

A …….: Imputável *(Accountable)* C …….: Consultado

R …….: Responsável I ……...: Informado

4.2.3. Práticas de desenvolvimento de *software*

As práticas de desenvolvimento de *software* são o alicerce de qualquer metodologia. Nelas é possível identificar os princípios orientadores ou até mesmo diretivos da forma de trabalho, centralizando e uniformizando os métodos de trabalho.

Um dos objetivos da aplicação de práticas transversais ao projeto é ter uma linha condutora ao longo do projeto de desenvolvimento de *software*.

No âmbito da presente dissertação, e conforme referido anteriormente, pretende-se que a metodologia proposta seja estruturalmente orientadora de modo a permitir que as práticas a implementar sejam adotadas e compreendidas por toda a equipa. Contudo, no que diz respeito ao desenvolvimento de *software per si*, pretende-se que a equipa de desenvolvimento tenha autonomia para internamente efetuar escolhas.

Neste sentido, são apresentadas um conjunto de práticas de caráter mandatório e um conjunto de práticas facultativas.

O subcapítulo 4.2.3.1. identifica um conjunto de práticas relacionadas com os aspetos de desenvolvimento de *software* onde foram identificadas lacunas e que se pretendem melhorar/corrigir (e.g. Frequência dos entregáveis, Comunicação da equipa com o IGFEJ, Documentação necessária, etc). As práticas de caráter facultativo relacionam-se com a organização interna da equipa de desenvolvimento.

4.2.3.1. Práticas de caráter mandatório

As práticas incluídas na presente metodologia proposta são as seguintes:

Sprint – O desenvolvimento por *sprints* tem como objetivo a entrega incremental e iterativa do produto. Um *sprint* consiste no desenvolvimento de um conjunto de requisitos definidos na listagem dos mesmos, rematando com os entregáveis correspondentes. O período de cada *sprint* não deve ultrapassar um mês de trabalho.

Planeamento dos *sprints* – O planeamento dos *sprints* ocorre num conjunto de reuniões onde são planeados e definidos todos os *sprints* do projeto. É um processo que engloba a estimativa de tempo de implementação dos requisitos anteriormente levantados e a distribuição dos mesmos pelos *sprints*.

Revisão e teste do *sprint* – A revisão e teste do *sprint* passa pela demonstração dos resultados obtidos por parte da equipa de desenvolvimento e pelo teste e validação dos mesmos por parte do gestor interno, bem como pelo cliente. Caso exista uma equipa de *testers* a mesma deve reportar nesta fase.

Avaliação do projeto – A avaliação do projeto consiste na análise e reflexão dos desenvolvimentos já efetuados. É neste período que a introdução ou alteração dos requisitos pode ser inserida no projeto. Esta avaliação é feita pelo gestor interno e pelo cliente.

4.2.3.2. Práticas de caráter facultativo

As práticas de caráter facultativo estão relacionadas com a gestão interna da equipa e a mesma pode ou não seguir, consoante ache mais conveniente no desenvolvimento do projeto.

Reunião diária – Nesta reunião devem ser apresentados os objetivos diários de cada membro, bem como, os problemas ou dificuldades que cada um está a enfrentar de modo a ultrapassá-los em conjunto.

Introspeção do *sprint* – Esta prática tem como objetivo efetuar uma reflexão sobre o trabalho desenvolvido, de modo a que sejam detetados os problemas que surgem, corrigindo-os, para que nas novas iterações o mesmo não se repita. Nesta reunião devem estar presentes os membros da equipa de desenvolvimento.

4.2.4. Fluxo de informação

A componente do fluxo de informação pretende clarificar de que maneira todas as componentes se interligam. Consiste na definição de como será transmitida, quem irá receber, quando será passada, e como vai ser entregue a informação gerada no decorrer do projeto.

Os *stakeholders* intervenientes nesta dimensão são semelhantes aos apresentados na subsecção da responsabilização das tarefas, 4.1.2. No fluxo de informação são também apresentadas duas soluções, uma para cada tipo de projeto (projeto interno e chave-na-mão).

O fluxo de informação passa por sete fases de desenvolvimento comuns aos dois tipos de projetos (Figura 4), a saber:

Levantamento de requisitos – O levantamento de requisitos consiste na execução de um conjunto de reuniões, preferencialmente presenciais, onde são debatidos e acordados os requisitos de desenvolvimento do projeto, bem como a priorização dos mesmos. O período de reuniões não deve ultrapassar os cinco dias úteis de trabalho e os intervenientes deverão ser o gestor interno e o cliente.

Tem como *input* a informação disponível sobre o projeto (preferencialmente deve ser entregue o 'Project Charter' preenchido) e gera como *output* o 'Documento de levantamento de requisitos' e a 'Análise Funcional' (caso seja adequada).

Planeamento dos *Sprints* – Esta fase consiste na definição e construção de todos os *sprints* de trabalho do projeto. É importante contemplar, durante a definição e planeamento dos *sprints*, o tempo necessário na execução das fases de revisão e replaneamento de *sprint* e avaliação de projeto. O período de implementação de *sprint* não deve ultrapassar um mês de desenvolvimento. Adicionalmente, é nesta fase que se preenche o documento de 'Plano de Projeto'. Os *outputs* gerados são o 'Plano de Projeto' e 'Lista de Requisitos' associados a cada *sprint*. Estes *outputs* devem no máximo durar cinco dias úteis a serem gerados. Deve intervir nesta fase o(s) gestor(es) de projeto, a equipa de desenvolvimento e o responsável pelas infraestruturas.

Execução de *sprint* – Esta fase deve ter a duração do *sprint* correspondente e tem como objetivo desenvolver os requisitos associados ao mesmo.

Revisão e testes do *sprint* – Esta fase consiste numa reunião entre a equipa de desenvolvimento, o cliente e o(os) gestor(es) de projeto e tem como objetivo rever, testar e validar o *sprint* desenvolvido. Esta fase, tal como o planeamento de *sprint* e levantamento de requisitos, não deve ultrapassar os cinco dias úteis. Após a reunião, deve ser elaborado o documento de 'Progresso do Projeto' para que fiquem claros os requisitos validados e não validados.

Avaliação do projeto – A avaliação do projeto tem como objetivo a apreciação do projeto como um todo, devendo ser analisado o estado do mesmo, no sentido de mitigar eventuais riscos futuros. Esta fase consiste numa reunião entre o gestor interno e o cliente.

Replaneamento do *sprint* – Nesta fase pretende-se que sejam incluídas as alterações documentadas na fase anterior, na 'Listagem de Requisitos' e consequentemente que seja

efetuado o replaneamento dos *sprints*. Esta fase consiste numa reunião entre o(s) gestor(es) de projeto e a equipa de desenvolvimento.

Fecho do projeto – A fase de fecho de projeto passa pela reunião entre o(s) gestor(es) e a equipa de desenvolvimento, onde são discutidos os problemas que ocorreram durante o projeto e como os mesmos foram ultrapassados. O objetivo é fazer o balanço do sucedido no projeto, no sentido de não voltar a cometer os mesmos erros futuramente.

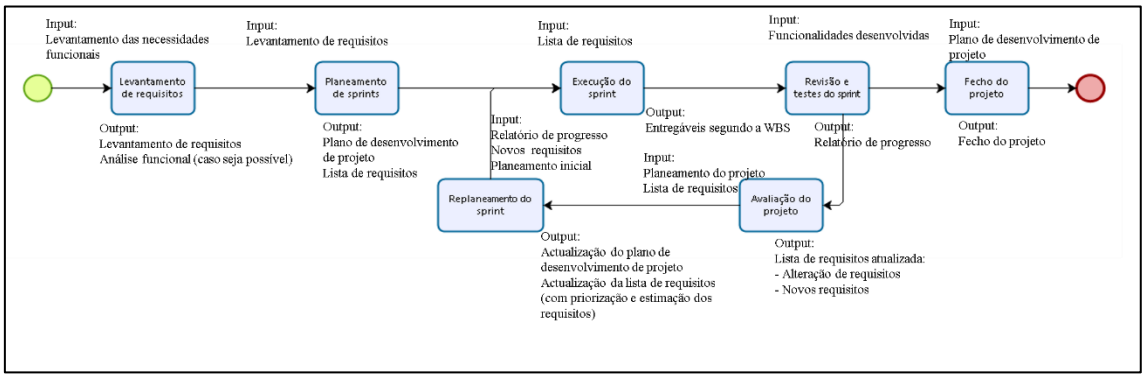


Figura 4 - Fluxo de informação para ambos os tipos de projetos

Apesar das sete fases expostas anteriormente serem comuns a ambos os tipos de projetos, os papéis de cada interveniente em cada uma delas não é. A Figura 5 apresenta as tarefas/funções de cada interveniente nas respetivas fases de desenvolvimento do tipo de projetos internos, de modo a que cada recurso tenha consciência das suas responsabilidades. Relativamente aos projetos chave na mão, a diferença é a existência de um gestor externo de projeto. A presença desta entidade resulta numa alteração de responsabilidades e tarefas dos restantes intervenientes demonstradas na Figura 6.

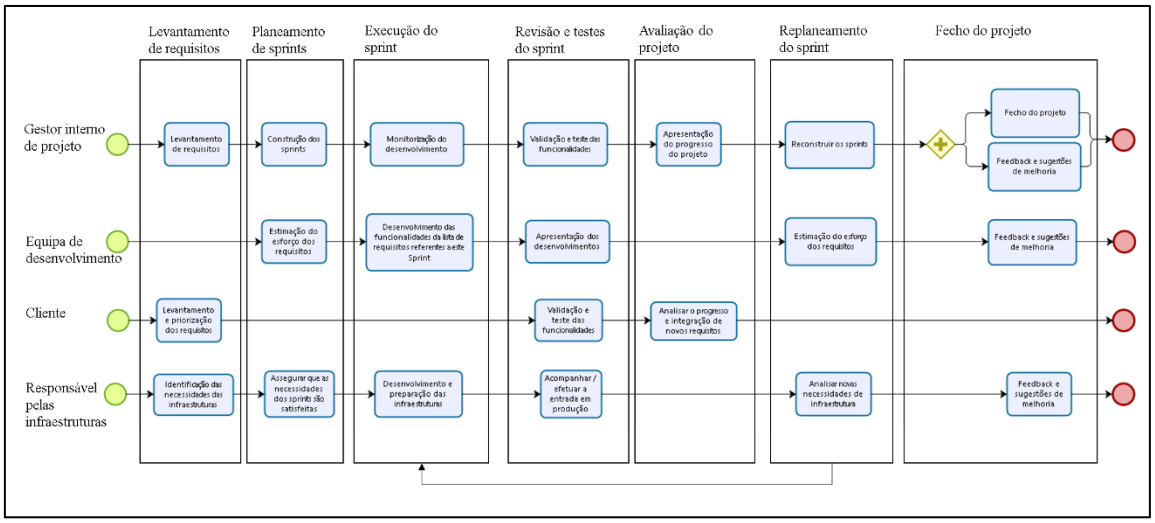


Figura 5 - Fluxo de informação detalhado dos projetos do primeiro tipo

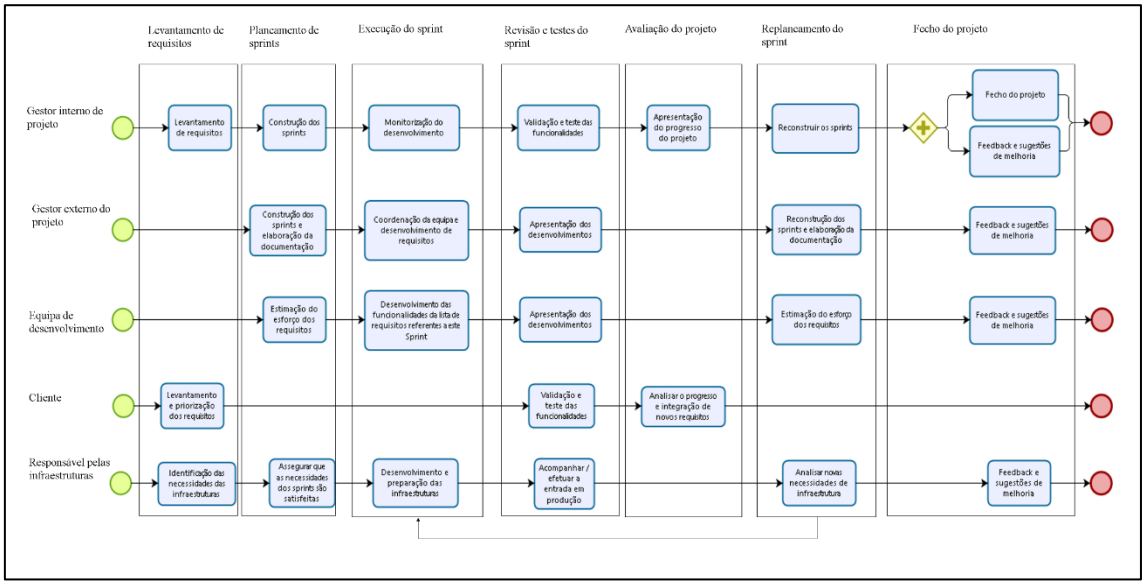


Figura 6 - Fluxo de informação detalhado dos projetos do segundo tipo (chave na mão)

Capítulo 5 – Implementação do modelo e análise dos resultados

Tal como foi enunciado no primeiro capítulo, a validação da adequação do modelo criado no âmbito da presente dissertação passa pela implementação do mesmo num projeto real. Este capítulo aborda todos os aspetos referentes a essa implementação e aos resultados obtidos com a mesma num projeto do IGFEJ. O mesmo subdivide-se em dois subcapítulos referentes às duas componentes que constituem o modelo, isto é, a aplicação das estratégias de mitigação e aplicação da metodologia criada num projeto real. Relativamente à metodologia de investigação (*Design Science Research*) referida no capítulo 1.4, este capítulo integra a fase de 'Avaliação do Artefacto'.

Tal como foi referido no capítulo 4, a metodologia passou por uma fase subsequente de maturação, tendo sido revista após a implementação das estratégias de gestão da mudança, considerando-se o *feedback* dos colaboradores do IGFEJ.

Nesse sentido, a metodologia mencionada no subcapítulo 5.1 refere-se à primeira versão (anterior à fase de maturação) e a exposta nos subcapítulos subsequentes refere-se à segunda versão, após o *feedback* dos colaboradores.

5.1. Análise de resultados da implementação das estratégias de gestão da mudança

A implementação das estratégias de gestão da mudança foi efetuada através de um *workshop* constituído pelas três medidas de mitigação abordadas no subcapítulo 4.2: 1ª formação, 2ª exercício prático e 3ª avaliação. O *workshop* teve lugar nas instalações do IGFEJ numa das suas salas de formações e contou com a participação de 16 colaboradores, cerca de 90% dos gestores de projeto do IGFEJ que cobrem todos os departamentos de sistemas de informação da organização. Apesar de todos os participantes terem estado presentes na primeira e segunda fase do *workshop*, apenas 11 responderam ao questionário correspondente à terceira fase. Esta última, ao contrário das duas primeiras, não foi finalizada no *workshop*. Considerou-se que faria sentido que os participantes refletissem antes de passar para a fase de avaliação da metodologia. Foi acordado um prazo de cinco dias para o preenchimento do questionário sendo que apenas 11 dos 16 participantes o preencheram.

As perguntas do questionário foram as seguintes:

Q1: Considera que existe a necessidade da adoção de uma metodologia de desenvolvimento de *software* nesta organização?

Q2: Considera que a metodologia apresentada é apropriada no contexto desta organização?

Q3: Classifique a relevância dos documentos da metodologia apresentada relativamente ao desenvolvimento de *software* na organização?

Q4: Classifique a adequação dos seguintes documentos nas necessidades de gestão de projeto da empresa?

Q5: Avalie os seguintes documentos quanto à facilidade no preenchimento (Lista de requisitos, Relatório de progresso, Plano de projeto, *Project Charter*, Requisito de mudança)?

Q6: Classifique a relevância das seguintes fases/práticas no desenvolvimento de software da organização (Levantamento de requisitos, Planeamento de *Sprints*, Execução do *Sprints*, Revisão e testes do *Sprints*, Replaneamento do *Sprint*)?

Q7: Classifique a adequação das seguintes fases/práticas nas necessidades de gestão de projeto da empresa?

Q8: Considera que a metodologia apresentada traz benefícios à organização?

Q9: Considera que a metodologia torna o desenvolvimento de *software* da organização mais eficiente?
Q10: Considera que a metodologia correspondeu às suas expectativas?
Q11: Estaria disposto a utilizar esta metodologia no futuro?

As questões estão divididas em dois tipos: Escala ordinal com cinco níveis (ordenação ascendente) - Q3, Q4, Q5, Q6 e Q7; Resposta sim-talvez-não - Q1, Q2, Q8, Q9, Q10 e Q11.
Os resultados das questões com resposta sim/talvez/não encontram-se na tabela 8:

Questões	Número de respostas "Sim"	Número de respostas "Talvez"	Número de respostas "Não"	Percentagem de respostas "Sim"	Percentagem de respostas "Talvez"	Percentagem de respostas "Não"
Q1	8	3	0	73%	27%	0%
Q2	6	3	2	55%	27%	18%
Q8	6	5	0	55%	45%	0%
Q9	6	4	1	55%	36%	9%
Q10	10	0	1	91%	0%	9%
Q11	6	5	0	55%	45%	0%

Tabela 8 – Resultados obtidos às questões com resposta sim/talvez/não

Em congruência com esta informação estão os resultados da adequação dos documentos. Relativamente à relevância dos documentos verificou-se, com base nos resultados obtidos na Figura 7, que o mais relevante é a 'Lista de Requisitos' seguido pelo 'Plano de Projeto'. Neste caso a 'Lista de Requisitos' e 'Plano de Projeto' também são considerados os mais apropriados. Pelo contrário, destacando-se com 2.7 valores em termos de relevância e com 3.3 em termos de adequação apresenta-se, respetivamente, o 'Project Charter' e o 'Requisito de mudança'.

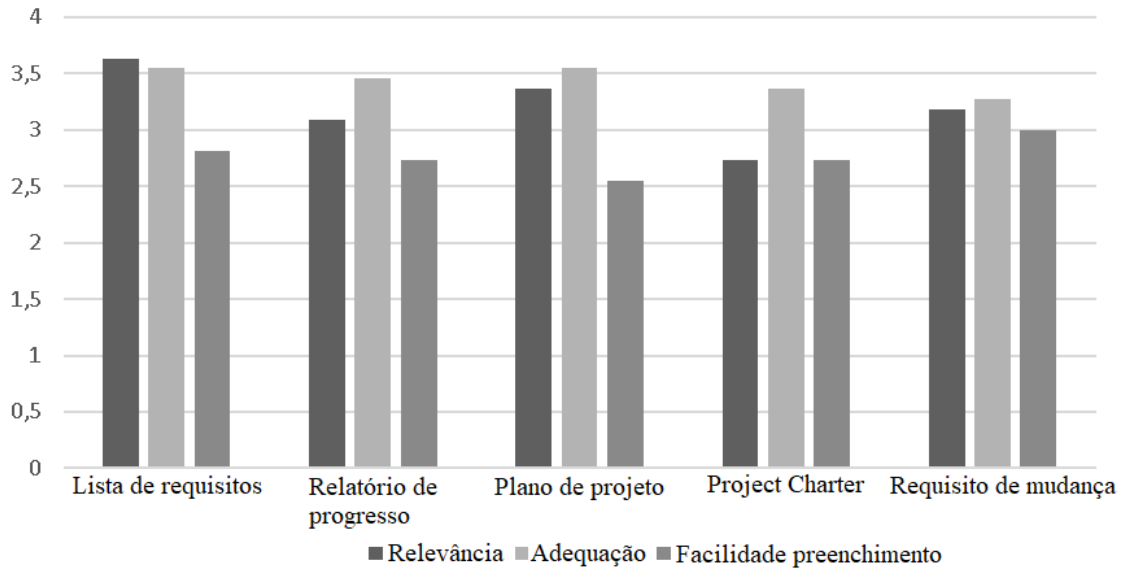


Figura 7 – Grau de relevância e adequação e facilidade dos documentos da metodologia

Este último, apesar de ser considerado um dos documentos menos relevantes, foi apontado o mais fácil de preencher. Relativamente às práticas, com base nos dados apresentados na Figura 8, foram consideradas em média, todas relevantes com 3.5 valores. O mesmo ocorreu na questão da adequação das práticas ao contexto da organização cuja média é de 3.3 valores.

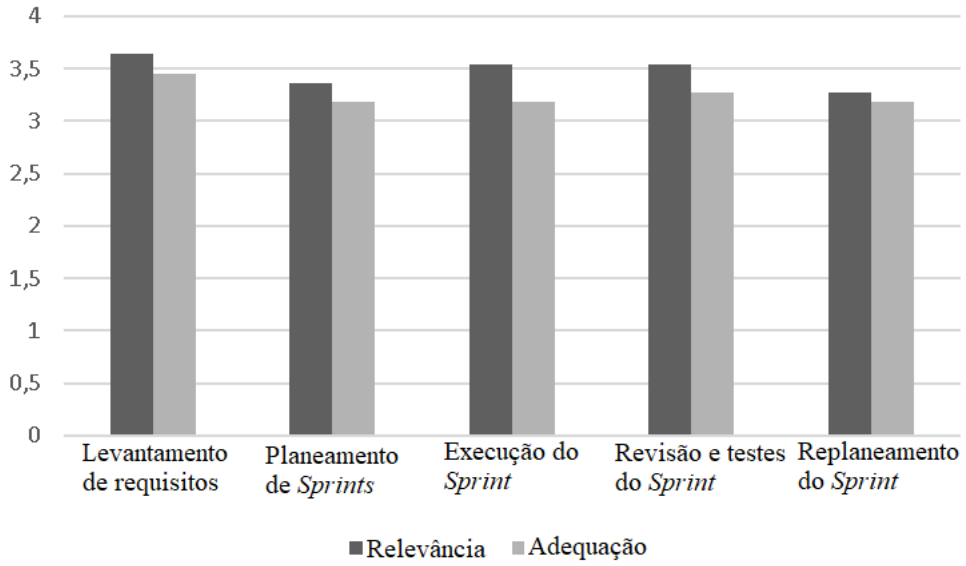


Figura 8 – Grau de relevância e adequação das práticas/fases da metodologia

Aproximadamente 73% dos participantes consideraram que existe a necessidade de adoção de uma metodologia de desenvolvimento de *software*, sendo que, os restantes consideraram que talvez fosse necessário, não havendo nenhuma resposta negativa. Tal

também acontece com as questões relativas aos benefícios, eficiência e predisposição de utilização da metodologia, todas com 55% de respostas positivas. Nestes casos, apenas a eficiência do desenvolvimento de *software* na organização teve uma resposta negativa (9%) e 36% de respostas 'Talvez'. As restantes duas questões não tiveram nenhuma resposta negativa e acabaram com 45% de respostas 'Talvez'.

Relativamente à adequação da metodologia ao contexto da organização, 55% dos colaboradores consideraram que se adequa e 27% consideraram que talvez se adequasse. Apenas 18% (dois participantes) não consideraram que se adequa ao contexto.

Para além destes resultados foi incluído um espaço para sugestões de melhoria onde foi retirado o seguinte *feedback*:

- Deveria ser incluída uma fase de testes mais explícita;
- O cliente deveria ser mais responsabilizado por algumas das fases e decisões;
- A infraestrutura deveria ter um papel mais presente no ciclo de vida apresentado;
- Deveria ser possível incluir requisitos não funcionais, técnicos e do projeto.

Tal como foi apresentado anteriormente, a criação da metodologia de desenvolvimento ágil de software passou por duas fases. A segunda fase teve como fundamento o *feedback* apresentado anteriormente. Foram acrescentadas fases ao ciclo de vida de desenvolvimento de *software*, alteradas rubricas nos documentos e responsabilidades aos *stakeholders* no projeto, de maneira a complementar e adaptar a metodologia às necessidades levantadas. Estas alterações estão contempladas na definição da metodologia exposta no subcapítulo 4.2.

5.2. Descrição da implementação da metodologia no projeto do IGFEJ

O seguinte subcapítulo descreve a implementação da metodologia proposta no projeto do IGFEJ. No sentido de permitir fazer uma descrição e consequente análise de dados fidedigna, o projeto foi acompanhado presencialmente, desta forma não só foi possível efetuar a descrição e análise, como apoiar no processo de adopção da metodologia.

O projeto surgiu de uma necessidade identificada pelo Ministério da Justiça representado por um grupo de trabalho (cliente) constituído por alguns elementos do Ministério e por futuros utilizadores finais da plataforma. O 'Project Charter' proposto na presente metodologia não foi desenvolvido, já que as necessidades foram apresentadas num documento com características semelhantes.

O projeto teve como *kick-off* o dia onze de Junho de 2018 e como data final, isto é, entrada em produção, dia dezanove de Novembro de 2018. Dado que a data de entrega da seguinte dissertação é anterior à data de finalização do projeto, não foi possível acompanhar o projeto até ao seu término. O último relatório de *input* data de dezoito de Outubro de 2018. A reunião de *kick-off* teve como propósito apresentar todas as componentes do projeto às equipas, incluindo a metodologia de desenvolvimento de *software* a adotar no mesmo. Nesta reunião estiveram presentes todos os *stakeholders* do projeto, inclusivamente um representante da administração do IGFEJ que reforçou a importância da aplicação da metodologia neste projeto na solução dos problemas atualmente existentes no desenvolvimento de *software* do IGFEJ.

Na reunião foram ainda esclarecidas todas as dúvidas relativas ao modo de funcionamento da metodologia mitigando alguma resistência ainda presente à utilização da mesma.

O gestor de projeto nomeado para este projeto tem várias responsabilidades na organização, sendo que a sua prioridade é garantir o bom funcionamento de um sistema de informação já existente, nesse sentido a sua disponibilidade para a gestão deste projeto não era 100%.

O projeto é composto por três equipas internas e tem a coordenação de um gestor acima referido. As equipas que constituem o projeto são: Equipa de desenvolvimento de *back-end*, equipa de desenvolvimento de *front-end,* e equipa que irá desenvolver uma componente específica do projeto num *software* também ele particular.

As equipas trabalham separadamente, mas a interação entre elas é diária, já que coexistem todas no mesmo local de trabalho e sobre a mesma chefia. Em paralelo ao desenvolvimento de *software,* foi criada uma equipa de infraestruturas que assegura a operacionalização das redes, sistemas e base de dados do projeto. Com base nestes dados,

o projeto foi identificado como projeto interno (primeiro tipo de projeto apresentado no capítulo 3).

Por questões de confidencialidade, dois aspetos relativos ao projeto não podem ser mencionados sendo estes, âmbito e nome.

O projeto foi dividido em seis *sprints* definidos na fase de 'Planeamento de Sprints'. Nesta fase foram também distribuídos os requisitos e estabelecidos os entregáveis a implementar/entregar em cada um deles. Relativamente aos documentos, para além da listagem de requisitos foi, da mesma forma, elaborado o documento de 'Planeamento do Projeto'.

O planeamento inicial repartiu cronologicamente os *sprints* da seguinte forma:

- 1º *Sprint* - 06/07/2018 a 24/07/2018
- 2ª *Sprint* - 27/07/2018 a 14/08/2018
- 3º *Sprint* - 17/08/2018 a 04/09/2018
- 4º *Sprint* - 07/09/2018 a 25/09/2018
- 5º *Sprint* - 28/09/2018 a 15/10/2018
- 6º *Sprint* - 18/10/2018 a 09/11/2018

Na fase de 'Planeamento dos *sprints'* foi também definido um período de dois dias entre os mesmos para a concretização das fases contempladas na metodologia (i.e. 'Revisão e testes do *sprint'*, 'Avaliação do projeto' e 'Replaneamento do *sprint'*). Foram, da mesma forma, salvaguardados cinco dias para testes de aceitação e/ou para eventuais atrasos nas entregas. O planeamento dos períodos de cada *sprint* teve como fundamento a dificuldade, criticidade e quantidade de funcionalidades que cada um comporta.

Nos dois primeiros *sprints* a metodologia foi aplicada na sua totalidade. Foram elaborados dez documentos, dois referentes ao 'relatório do sprint' e os restantes referentes a alterações de requisitos (documento 'requisito de mudança'). Ainda referente aos documentos, dada a integração/alteração de requisitos no projeto, o documento 'plano de projeto' e a 'listagem de requisitos' foram atualizados, respetivamente, com o replaneamento dos sprints e com as novas funcionalidades a serem desenvolvidas. Relativamente às fases, todas elas foram cumpridas no decorrer dos dois primeiros *sprints*. É importante referir que na fase de 'avaliação do projeto' pertencente ao segundo *sprint* foi identificado que no seguinte os recursos iam diminuir para metade, uma vez que se iria iniciar o período de férias. Com esta informação foi efetuado um

replaneamento ao projeto de maneira a colmatar os atrasos previstos nos desenvolvimentos do terceiro *sprint.*

No terceiro *sprint,* para além da ausência de elementos da equipa (questão identificada anteriormente) evidenciou-se uma redução da presença da administração do IGFEJ no projeto. O acompanhamento efetuado nos dois *sprints* anteriores deixou de se verificar pela sua ausência nas reuniões e nas interações com o gestor de projeto.

Nos restantes *sprints* analisados, quarto e quinto, o foco do projeto centrou-se no desenvolvimento das funcionalidades, pelo que as restantes fases e documentos para além da 'execução do *sprint'* foram progressivamente escamoteadas. No quarto, apesar de terem sido identificados na fase de 'avaliação do projeto' alterações nos requisitos, os documentos de 'requisito de mudança' não foram elaborados. A par com este documento, o 'relatório de progresso' do projeto também não foi preenchido. No quinto, a utilização da metodologia continuou a diminuir já que, para além dos documentos 'requisito de mudança' e 'relatório de progresso' não terem sido preenchidos, as fases de 'avaliação do projeto', 'revisão e testes do *sprint*' e 'replaneamento do *sprint*' também não foram cumpridas. Para além do incumprimento das fases e dos documentos, os próprios entregáveis do sprint em questão também não foram cumpridos gerando atraso no planeamento.

A ausência da administração do IGFEJ no projeto, a par da utilização da metodologia, foi progressivamente denotada. As suas interações com o gestor de projeto foram escassas e a sua preocupação já não se focava no incentivo à utilização da metodologia, mas na evolução funcional do projeto.

A interação com o cliente (grupo de trabalho do Ministério) foi regular ao longo dos quatro primeiros *sprints*. No último avaliado (quinto), a comunicação foi efetuada *ad hoc* em reuniões com o objetivo da apresentação, pelos clientes, de novas funcionalidades e para a aprovação dos desenvolvimentos efetuados.

Tal como foi exposto no início do presente capítulo, o projeto não foi acompanhado até à sua conclusão, pelo que não foi possível verificar se a fase 'Fecho do Projeto' foi executada, bem como se o último *sprint* foi desenvolvido como esperado ou se foram registados atrasos.

5.3. Análise crítica dos resultados da implementação da metodologia no projeto do IGFEJ

Considerando os dados referidos no subcapítulo anterior que descrevem a implementação da metodologia criada num projeto do IGFEJ , foi analisado o impacto que a mesma teve na organização, bem como os fatores que levaram ao seu sucesso/insucesso.

Como resultado dessa análise verifica-se claramente, que a implementação da metodologia atravessou dois momentos. Um primeiro onde se identifica a implementação da metodologia na sua totalidade respeitando todas as fases de desenvolvimento, documentação e responsabilidades atribuídas. E uma segunda fase onde se evidencia uma diminuição substancial na utilização e cumprimento da metodologia ágil.

Através da análise efetuada ao projeto no período de transição do primeiro momento para o segundo, foi possível identificar duas causas para a sua ocorrência:

- Redução no apoio e incentivo por parte da administração do IGFEJ na utilização da metodologia;
- Falta de capacidade e de motivação do gestor de projeto em manter as práticas seguidas até então.

Apesar de num primeiro momento, se ter observado *outputs* positivos nos resultados da aplicação da metodologia (e.g. inclusão de novos requisitos no projeto sem afetar os desenvolvimentos, análise e mitigação de riscos através de uma avaliação contínua do planeamento e das variáveis externas ao projeto) os mesmos não foram suficientes para a adoção voluntária da metodologia que resultou no seu desuso no decorrer do projeto.

O facto de a metodologia não ter sido cumprida até ao término do projeto, levou a que o atraso observado no final no quinto *sprint* não tivesse, até ao momento de entrega da dissertação, resultado num replaneamento do projeto. Nesse sentido, a falha de um ou mais fatores que permite identificar o sucesso do projeto segundo o PMBOK (entrega do âmbito, no tempo e custo definidos), torna-se num risco real ao qual a organização fica sujeita.

Nesse sentido verifica-se que as estratégias de gestão da mudança aplicadas na organização, apesar de mitigarem parte dos problemas (Utilidade percebida, Facilidade de utilização percebida, Compatibilidade percebida, Maturidade percebida e Falta de conhecimento) identificados noutros estudos empíricos, não foi suficiente para o contexto em questão. A sensibilização da gestão de topo não foi efetuada com sucesso levando a que a adoção da metodologia tivesse o mesmo destino. Para a aplicação de uma

metodologia ágil num contexto resistente como este, é necessário haver uma incidência maior na sensibilização das chefias das organizações.

Neste caso a eleição do gestor de projeto teve impacto em todo o projeto. Verificou-se que o facto de ser par (em outros projetos) de alguns membros da equipa que geria, aliada à falta de disponibilidade que tinha no foco deste projeto, foram dois dos fatores que levaram à incapacidade do gestor fazer prevalecer a metodologia.

No decorrer da análise à implementação da metodologia, para além das questões relativas à sua adoção por parte dos colaboradores, identificaram-se através do *feedback* dos mesmos e através do acompanhamento do projeto, algumas lacunas que devem ser consideradas em novas iterações.

- As decisões tomadas ao longo do projeto foram documentadas na 'listagem de requisitos' (classificadas como requisito de projeto). Isto levou a que após a população do documento com os requisitos, a informação tornou-se difícil de encontrar;
- Outra questão que se levantou relativamente a este documento, foi as várias versões que o mesmo acabou por possuir ao fim dos dois primeiros *sprints*. Dada a quantidade de equipas e, por conseguinte, a quantidade de cópias do ficheiro, tornou-se difícil que o mesmo estivesse atualizado em todas as suas instâncias;
- O documento 'Mudança de Requisito' também demonstrou falta de eficiência pois não era possível, de forma intuitiva, identificar o documento correspondente ao requisito que se queria analisar. Este facto ocorreu devido aos nomes dos ficheiros que não permitiam fazer essa distinção. Esta questão foi resolvida em pleno projeto ao identificar o documento com algumas palavras chave do requisito em questão (e.g. "requisito mudança_número do requisito_ palavras chave");
- Outra lacuna identificada na metodologia foi a inexistência de uma rubrica que permitisse a inclusão do desenho da arquitetura do documento.

Em suma, é possível destacar valor acrescentado bem como espaço de melhoria na adoção das práticas propostas. No contexto do projeto do IGFEJ, enquanto que numa primeira fase a metodologia serviu como base para a estruturação do trabalho a desenvolver, numa segunda fase foi negligenciada, tanto em aspetos mais funcionais (como a não elaboração da documentação), bem como em aspetos culturais (como a resistência à sua adoção).

Capítulo 6 – Conclusões

6.1. Principais conclusões

A utilização das metodologias ágeis no desenvolvimento de *software* nas organizações estão, efetivamente, a emergir. Este aumento deve-se principalmente, ao sucesso que apresentam na entrega dos projetos. Em 2017, cerca de 28% dos projetos desenvolvidos com metodologias ágeis são mais bem sucedidos que os desenvolvidos com metodologias tradicionais. Em 2015, averiguou-se que 39% dos projetos elaborados com metodologias ágeis tiveram sucesso enquanto que efetuados com metodologias tradicionais tiveram apenas 11% de sucesso. Estes dados justificam a maior utilização das metodologias ágeis face às tradicionais (Standish Group 2015).

Relativamente ao estudo efetuado na relação entre o paradigma Lean e Ágil verificou-se que não existe consenso entre os diferentes autores que contribuíram para a comunidade científica. Nesse sentido, o foco do presente estudo focou-se nas metodologias ágeis e nos seus impactos nos projetos e organizações.

Sobre as diferentes metodologias ágeis apurou-se que o Scrum e o XP são atualmente, as metodologias mais usadas no desenvolvimento de *software*. Este fenómeno foi também identificado nos projetos desenvolvidos com este tipo de metodologias no sector público. Na verdade, foram apenas encontrados estudos empíricos referentes à implementação do Scrum em empresas neste sector.

A presente dissertação centrou-se na aplicação e análise dos impactos de uma metodologia ágil de desenvolvimento de *software* numa empresa do sector público. O trabalho desenvolvido focou-se na criação de um modelo que simplifique e facilite a aplicação da metodologia ágil num projeto real do IGFEJ.

Verificou-se que a metodologia desenvolvida quando aplicada no projeto (durante os dois primeiros *sprints*) obteve *outputs* vantajosos e impactantes na organização através da descoberta atempada de riscos permitindo mitigar os mesmos. A metodologia adequou-se ao contexto e às necessidades do desenvolvimento de *software* da organização, permitindo que o projeto se desencadeasse de forma organizada, eficiente e eficaz não só através do cumprimento dos prazos estabelecidos, como na possibilidade de inclusão de novos requisitos sem afetar os elementos da equipa e os clientes.

Os problemas identificados no âmbito da aplicação da metodologia (evidenciados após os dois primeiros *sprints*), permitem enumerar sugestões de melhoria apresentadas no subcapítulo 6.2.

É importante referir que o acompanhamento do projeto ocorreu durante quatro meses e meio. Em cerca dos dois primeiros meses, a metodologia foi aplicada na íntegra. Após esse período, surgiram falhas na sua adoção com consequentes atrasos no projeto.

O horizonte temporal em que a adoção da metodologia foi cumprida assumiu-se como diminuto, uma vez que não permitiu apurar de forma abrangente o impacto que a adoção de novos métodos têm no projeto e na organização.

Apesar disso, e tendo como pressuposto o curto horizonte temporal, o maior impacto que se verificou na aplicação da metodologia foi, de facto, a resistência à mesma. Ao comparar os dados recolhidos nos questionários do *workshop*, onde a maioria dos inquiridos consideraram que é necessário a adoção de uma metodologia ágil no IGFEJ, que a metodologia apresentada acresce benefícios e eficiência no desenvolvimento e que se demonstraram predispostos a utilizar a mesma, no presente exemplo prático isso não se verificou. Pode-se concluir que os estudos teóricos podem não se refletir na realidade, tal como se verificou neste caso. Apesar de se ter tentado mitigar a resistência à mudança não só através do *workshop* mas em todo o processo de implementação da metodologia (i.e. reunião de kick off e esclarecimento de dúvidas), essa resistência prevaleceu aquando a introdução de novos métodos levando a que a adoção dos mesmos não resultasse na sua totalidade e abrangência.

Analisando no estado de arte (capítulo 2) as principais causas da resistência à mudança nas organizações (quando em contacto com métodos ágeis), observa-se que uma das razões enumeradas é também evidenciada no âmbito da presente dissertação - dificuldade em convencer a equipa de topo a investir no novo método e falta de capacidade para alterar a cultura organizacional o que inclui a ausência de suporte dos gestores de topo.

No seguimento do que foi apresentado anteriormente e no sentido de clarificar o que se obteve relativamente às questões de investigação, pode-se concluir que de facto as metodologias ágeis podem ter impacto na melhoria do desenvolvimento de *software* quando bem implementadas mas que, para tal, é necessário haver um esforço conjunto de toda a empresa para que os métodos sejam implementados e adotados pelos colaboradores. Com base nos resultados obtidos observa-se que a metodologia ágil apresentada pode ser uma opção viável para as organizações do sector público pelo que quando aplicada trouxe não só alinhamento entre toda a equipa, mas também benefícios em termos de análise de riscos e cumprimento de *deadlines*.

Em resumo podem ser retiradas três principais contribuições do estudo relacionadas com a implementação da metodologia:

1. Para colmatar a resistência à mudança deve haver uma maior incidência na sensibilização dos gestores de topo;
2. A metodologia aplicada tem espaço para melhorar através das sugestões de melhoria apresentadas no subcapítulo seguinte;
3. Quando aplicada, a metodologia teve *outputs* positivos através de mitigação de potenciais atrasos nos desenvolvimentos, identificando atempadamente, riscos para o projeto, bem como permitiu a inclusão de requisitos no decorrer do projeto não afetando os desenvolvimentos da equipa e deixando os clientes satisfeitos.

6.2. Sugestões de melhoria

No seguimento da análise efetuada, dos problemas identificados e no sentido de complementar o trabalho com soluções que mitiguem os mesmos, foi efetuada uma reflexão que resultou num conjunto de sugestões de melhoria na aplicação da metodologia e nas estratégias de gestão da mudança.

No que diz respeito às estratégias de mitigação da resistência, existem duas questões que devem ser melhoradas. Uma relacionada com a sensibilização da chefia, nomeadamente e com maior incidência sobre o gestor de projeto que irá ser o catalisador da mesma, e outra relacionada com os elementos da equipa. Estes devem estar conscientes do valor acrescido da adoção de uma metodologia, compreendendo as vantagens no âmbito do seu trabalho de desenvolvimento. Desta forma, o método de trabalho não será visto como uma imposição ou algo que tem de ser seguido. A equipa tem a possibilidade de sentir que faz parte de uma mudança que acrescenta valor ao seu trabalho. Para integrar a equipa nestas mudanças, a comunicação deve passar também por expor as vantagens percebidas das metodologias, nomeadamente das metodologias ágeis para que a sua adoção seja mais agilizada. Para obter estes resultados sugere-se a complementação do workshop com casos práticos e ou estudos empíricos em que se verifica efetivamente um melhoramento do desenvolvimento de *software* não só para o cliente mas sobretudo para o elemento da equipa. Para além disso, sugere-se também que num cenário temporalmente mais alargado, a metodologia seja introduzida gradualmente na organização em detrimento da aplicação integral num projeto "protótipo". Desta forma, a mudança iria ser gerida progressivamente o que possibilita a diminuição da resistência à mesma.

No caso das chefias deve, de igual forma, fazer-se uma sensibilização mais focada nos mesmos. Para tal sugere-se a escolha meticulosa do gestor de projeto (preferencialmente uma pessoa que seja aberta a novos desafios).

No que diz respeito às práticas adotadas na metodologia, tal como identificado anteriormente, a ‘listagem de requisitos’ surgiu como o documento menos apto, podendo tornar-se confuso e complexo após conter grandes quantidades de informação. Para além disso, por ser um documento volátil dificulta o alinhamento de todos os intervenientes no desenvolvimento, nomeadamente quando existe mais que uma equipa presente no projeto. Nesse sentido, e para colmatar estes problemas, numa nova iteração da metodologia sugere-se a utilização de um *software* de gestão de desenvolvimento de *software*. Este tipo de ferramentas permite gerir os requisitos do projeto de maneira mais adequada e

eficaz através da centralização e disponibilização atualizada da informação (e.g. documentos, requisitos, decisões, avisos, etc). Por um lado, possibilita as equipas a gerir o esforço efetuado ao longo do projeto e por outro, aos gestores analisar e avaliar o progresso do mesmo. Dependendo da ferramenta poderá trazer outro tipo de vantagens para o projeto como versionamento do código, compilação e *deploy* de soluções, testes, etc.

Relativamente ao desenho da arquitetura, problema identificado no início do projeto, considera-se que a rubrica a incluir este conteúdo deve ser criada no documento 'plano de projeto'.

As sugestões de melhoria apresentadas resultam de uma análise crítica ao modo como foi implementada a metodologia proposta e adotada pelos colaboradores da organização. O exercício efetuado no âmbito da presente dissertação integra *outputs* concretos que pretendem não só introduzir valor na comunidade científica, bem como servir de base de documentação para trabalho futuro.

6.3. Trabalho futuro

Relativamente ao trabalho futuro no âmbito da implementação de metodologias ágeis no sector público sugere-se que o foco seja feito em torno das estratégias de gestão da mudança com especial foco na sensibilização da gestão de topo, no sentido de permitir uma adoção agilizada e bem sucedida da metodologia e de maneira a gerar o menor desconforto possível na organização.

Será também interessante perceber se o facto de o trabalho apresentado ter sido feito no âmbito académico, influenciou de alguma forma os resultados obtidos. Será que uma presença mais empresarial e menos académica sustentada em resultados comprovados, teriam surtido outro tipo de efeito no modo como os colaboradores abordaram a metodologia?

Outro ponto que deve ser explorado no futuro é a implementação da metodologia criada, após aplicadas as sugestões de melhoria, em mais projetos para que se possa através de uma amostragem maior, extrair mais informação que suporte as conclusões apreendidas.

Seria também interessante perceber como é que a metodologia se comportaria noutros contextos para além do público. É uma metodologia que apesar de ter sido construída a pensar nas organizações mais conservadoras poderá ser aplicada e ter resultados positivos noutros sectores.

6.4. Principais contribuições para a comunidade científica

As principais contribuições da presente dissertação para a comunidade científica são os *outputs* gerados no capítulo 4, onde é criado e apresentado o modelo de implementação de uma metodologia ágil de desenvolvimento de *software* no sector público e no capítulo 5, onde são extraídos os resultados da implementação prática do modelo criado no contexto em questão. A possibilidade de escalar o conteúdo criado para outros contextos faz com que o trabalho desenvolvido acresça na mais valia dada na comunidade.

Para além da expectável validação do estudo através da aplicação do modelo num projeto real, esta pesquisa é também suportada pela validação, aceitação e publicação de um artigo científico na conferência CENTERIS dedicada à averiguação da importância dos sistemas de informação nas organizações:

- Ribeiro, Afonso, and Domingues Luísa. 2018. "Acceptance of an Agile Methodology in the Public Sector." *Procedia Computer Science* 138: 621–29.

Bibliografia

Abdalhamid, S., & Mishra, A. (2017). Factors in Agile Methods Adoption. *TEM Journal*, *6*(2), 416–421.

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and analysis. *Espoo, Finland: Technical Research Centre of Finland, VTT Publications*, 478. Retrieved from http://arxiv.org/abs/1709.08439%0Ahttp://jeffsutherland.org/whatsnew.html

Asnawi, A. L., Gravell, A. M., & Wills, G. B. (2011). Empirical investigation on agile methods usage: issues identified from early adopters in Malaysia. *Agile Processes in Software Engineering and Extreme Programming*, 192–207.

Barton, B. (2009). All-out organizational scrum as an innovation value Chain. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on* (pp. 1–6).

Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70–77.

Boehm, B. (2002). Get ready for agile methods, with care. *Computer*, *35*(1), 64–69.

Brown, T. (2001). Modernisation or failure? IT development projects in the UK public sector. *Financial Accountability & Management*, *17*(4), 363–381.

Chan, F. K. Y., & Thong, J. Y. L. (2009). Acceptance of agile methodologies: A critical review and conceptual framework. *Decision Support Systems*, *46*(4), 803–814.

Cockburn, A. (2002). *Agile software development* (Vol. 177). Addison-Wesley Boston.

Conboy, K. (2009). Agility from first principles: reconstructing the concept of agility in information systems development. *Information Systems Research*, *20*(3), 329–354.

Conboy, K., Coyle, S., Wang, X., & Pikkarainen, M. (2011). People over process: key people challenges in agile development.

Conboy, K., & Fitzgerald, B. (2004). Toward a conceptual framework of agile methods: a study of agility in different disciplines. In *Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research* (pp. 37–44).

Conboy, K., & Wang, X. (2007). Agile practices in use from an innovation assimilation perspective: a multiple case study.

Conforto, E. C., Salum, F., Amaral, D. C., da Silva, S. L., & de Almeida, L. F. M. (2014). Can agile project management be adopted by industries other than software development? *Project Management Journal*, *45*(3), 21–34.

Dall'Agnol, M., Janes, A., Succi, G., & Zaninotto, E. (2003). Lean Management — A Metaphor for Extreme Programming? [5]. In *Extreme Programming and Agile Processes in Software Engineering, XP 2003* (Vol. 4, pp. 26–32). Retrieved from

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9–10), 833–859.

Ebert, C., Abrahamsson, P., & Oza, N. (2012). Lean software development. *IEEE Software*, *29*(5), 22–25.

Edquist, C., Hommen, L., & Tsipouri, L. (2000). Policy implications. *Public Technology Procurement and Innovation*, 301–311.

Fontana, R. M., Fontana, I. M., da Rosa Garbuio, P. A., Reinehr, S., & Malucelli, A. (2014). Processes versus people: How should agile software development maturity be defined? *Journal of Systems and Software*, *97*, 140–155.

Gandomani, T. J., Zulzalil, H., Azim, A., & Ghani, A. (2014). How Human Aspects Impress Agile Software Development Transition and Adoption. *International Journal of Software Engineering and Its Applications*, *8*(1), 129–148.

Ganesh Jonnalagadda, Sarah Shafey, Wes Lynah. 2017. *Agile Project Delivery Confidence*. http://www.zdnet.com/article/forrester- (October 29, 2018).

Hajjdiab, H., & Al Shaima Taleb. (2011). Adopting Agile Software Development: Issues and Challenges. *International Journal of Managing Value and Supply Chains*, *2*(3), 1–10.

Hamed, A. M. M., & Abushama, H. (2013). Popular agile approaches in software development: Review and analysis. *Proceedings - 2013 International Conference on Computer, Electrical and Electronics Engineering: "Research Makes a Difference", ICCEEE 2013*, 160–166.

Henderson-Sellers, B., & Serour, M. K. (2005). Creating a dual-agility method: The value of method engineering. *Journal of Database Management*, *16*(4), 1.

Highsmith, J. (2013). *Adaptive software development: a collaborative approach to managing complex systems*. Addison-Wesley.

Ihlsoon Cho, Y.-G. K. (2002). Critical factors for assimilation of object-oriented programming languages. *Journal of Management Information Systems*, *18*(3), 125–156.

Ikonen, M., Kettunen, P., Oza, N., & Abrahamsson, P. (2010). Exploring the sources of waste in kanban software development projects. In *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on* (pp. 376–381).

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, *51*, 915–929.

Jack T. Marchewka. (2013). *Information Technology Project Management, 4th Edition*.

Jalali, S., & Wohlin, C. (2010). Agile practices in global software engineering - A systematic map. *Proceedings - 5th International Conference on Global Software Engineering, ICGSE 2010*, 45–54.

Jalali, S., Wohlin, C., & Angelis, L. (2014). Investigating the applicability of Agility assessment surveys: A case study. *Journal of Systems and Software*, *98*, 172–190.

Kaczorowska, A. (2015). Traditional and agile project management in public sector and ICT. In *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on* (pp. 1521–1531).

Karaj, A., & Little, J. (2013). Transforming a Public Sector Company: From Stone Age to Agile. In *Agile Conference (AGILE), 2013* (pp. 74–81).

Kiran, J., & V Rama, K. (2013). AGILE SOFTWARE DEVELOPMENT AND CHALLENGES, 125–129.

Lalsing, V., Kishnah, S., & Pudaruth, S. (2012). People factors in agile software development and project management. *International Journal of Software Engineering & Applications*, *3*(1), 117.

Larson, Erick W, and Clifford F Gray. 2015. "A Guide to the Project Management Body of Knowledge: PMBOK (®) Guide."

Lee, G., & Xia, W. (2010). Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility. *Mis Quarterly*, *34*(1), 87–114.

Lopez-Martinez, J., Juarez-Ramirez, R., Huertas, C., Jimenez, S., & Guerra-Garcia, C. (2016). Problems in the adoption of agile-scrum methodologies: A systematic literature review. *Proceedings - 2016 4th International Conference in Software Engineering Research and Innovation, CONISOFT 2016*, 141–148.

Madadipouy, K. (2015). An Examination and Evaluation of Agile Methodologies for Systems Development. *Australasian Journal of Computer Science*, *2*(1), 1–17.

Mahanti, A. (2006). Challenges in Enterprise Adoption of Agile Methods – A Survey. *Journal of Computing and Information Technology -CIT*, *14*(3), 197–206.

Mary, P., & Tom, P. (2006). Implementing Lean Software Development: From Concept to Cash. Addison Wesley, Boston, USA.

Meso, P. (2006). Contemporary Practices in Systems Development Agile Software Development : Adaptive Systems Principles and Software Development. *Contemporary Practices In Systems Development*, 19–30.

Moniruzzaman, A., & Hossain, D. (2013). Comparative Study on Agile software development methodologies. *arXiv Preprint arXiv:1307.3356*, *V*(3), 37–56.

Nuottila, J., Aaltonen, K., & Kujala, J. (2016). Challenges of adopting agile methods in a public organization. *IJISPM-INTERNATIONAL JOURNAL OF INFORMATION SYSTEMS AND PROJECT MANAGEMENT*, *4*(3), 65–85.

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

Papadopoulos, G. (2015). Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects. *Procedia - Social and Behavioral Sciences*, *175*, 455–463.

Parker, R., & Bradley, L. (2000). Organisational culture in the public sector: evidence from six organisations. *International Journal of Public Sector Management*, *13*(2), 125–141.

Paulk, M. C. (2001). Extreme programming from a CMM perspective. *IEEE Software*, *18*(6), 19–26.

Perera, G. I. U. S., & Fernando, M. S. D. (2007). Enhanced agile software development - Hybrid paradigm with LEAN practice. In *ICIIS 2007 - 2nd International Conference on Industrial and Information Systems 2007, Conference Proceedings* (pp. 239–243).

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Addison-Wesley.

Powner, D. (2012). Software Development: Effective Practices and Federal Challenges in Applying Agile Methods. United States Government Accountability Office, Washington, DC.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

Riemenschneider, C. K., Hardgrave, B. C., & Davis, F. D. (2002). Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *IEEE Transactions on Software Engineering*, *28*(12), 1135–1145.

Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. *IEEE Software*, *17*(4), 26–32.

Roberts, T. L., Gibson, M. L., Fields, K. T., & Rainer, R. K. (1998). Factors that impact implementing a system development methodology. *IEEE Transactions on Software Engineering*, *24*(8), 640–649.

Roses, L. K., Windmöller, A., & Carmo, E. A. do. (2016). Favorability conditions in the adoption of agile method practices for software development in a public banking. *Journal of Information Systems and Technology Management*, *13*(3), 439–458.

Strode, D. E. (2005). The agile methods: An analytical comparison of five agile methods and an investigation of their target environment. *Unpublished Master of Information Sciences (Information Systems), Massey University, Palmerston North*.

Standish Group. 2015. "State of the Software Development Industry."
https://www.infoq.com/articles/standish-chaos-2015 (October 29, 2018).

Sulaiman, N. L., Mahrin, M. N., & Yusoff, R. C. M. (2016). Influential Factors on the Awareness of Agile Software Development Methodology : A Systematic Literature Review. *Journal of Internet Computing and Services*, *170*(5), 11.

Sultan, F., & Chan, L. (2000). The adoption of new technology: the case of object-oriented computing in software companies. *IEEE Transactions on Engineering Management*, *47*(1), 106–126.

The Scrum Guide - Scrum Alliance. (2014). Retrieved December 15, 2017, from https://www.scrum.org/

Thomas, R, Niederman F, P. Y. (2017). Exploring the Role of Agile Approaches for the Management of Projects.

Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M. (2013). Agile in Public Administration: Oxymoron or reality? An experience report. In *CEUR Workshop Proceedings* (Vol. 1017, pp. 1–8).

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

What is Agile Software Development? | Agile Alliance. (n.d.). Retrieved December 15, 2017, from https://www.agilealliance.org/agile101/

Williams, L., & Cockburn, A. (2003). Guest Editors' Introduction: Agile Software Development: It's About Feedback and Change. *Computer*, *36*(6), 39–43.

Wisitpongphan, N., & Khampachua, T. (2016). Agile in public sector: Case study of dairy farm management projects. *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 1–5.

Anexos e Apêndices

Anexos

Anexo A

Scrum

Scrum é uma *framework* de gestão e controlo que trata de problemas complexos, trabalhando de maneira criativa e produtiva na entrega de produtos que se adequem, o mais possível, às necessidades do negócio ("The Scrum Guide - Scrum Alliance," 2014). A metodologia trabalha sobre um conjunto de práticas que são descritas abaixo.

A seguinte secção é retirada do livro "Scrum in Action" da edição de 2012.

Segundo Pham, Andrew e Phuong-Van, autores do livro "Scrum in Action", a equipa Scrum deve ser composta por elementos multifuncionais. A mesma deve ser formada por um ScrumMaster, um Dono do Produto e uma Equipa de Desenvolvimento com as competências necessárias para criar o produto. Tudo começa com a informação recolhida pelo responsável do projeto aos *stakeholders*, para que posteriormente seja elaborado uma lista de requisitos priorizada (também denominada *backlog).* A mesma é constituída não só por requisitos funcionais, mas também por problemas técnicos, documentação, resolução de bugs, etc. Esta recolha de requisitos deve ser feita de maneira intensiva um ou dois dias antes da reunião de planeamento e lançamento dos *sprints* (*sprint planning* e *release sprint*, respetivamente*).* É importante que o dono do produto partilhe a informação resultante do lançamento dos *sprints* com a equipa, já que a mesma terá de conhecer o produto antes das reuniões de planeamento. O objetivo da reunião de lançamento é que a equipa de Scrum identifique todos os entregáveis que o projeto deveria ter, bem como, as datas prováveis de entrega. Esta reunião, idealmente deveria ter a duração de 4 horas e devem ser geridos ciclos de quatro semanas por *sprint*.

As reuniões de planeamento, por outro lado, devem durar cerca de oito horas para *sprints* de quatro semanas, e quatro horas para *sprints* de duas semanas. Uma boa prática nas reuniões de planeamento de *sprint* é dividir a reunião em dois períodos iguais. No primeiro período devem-se discutir os requisitos, como casos de uso, para decidir a que *sprint* correspondem e qual o seu objetivo.

Durante a segunda parte da reunião, o foco deve-se virar para a identificação das tarefas e para a dedução do número de horas que levaria a transformar estas tarefas em potenciais entregáveis. Se não existir qualquer tipo de *software* de planeamento de tarefas, as

mesmas devem ser colocadas num quadro para uma fácil alocação e monitorização das tarefas.

Após a reunião de lançamento e de planeamento de *sprints*, começa efetivamente o trabalho do *sprint* com as reuniões diárias de quinze minutos.

Ao contrário das metodologias tradicionais onde o gestor de projeto é responsável pela organização semanal do estado do projeto, com o Scrum, a equipa reúne-se todos os dias para inspecionar, não o estado, mas o progresso da equipa no projeto tendo em vista o objetivo do *sprint*. No sentido de controlar este progresso, deve ser criado um gráfico que demonstre o trabalho feito e o que está em falta para a conclusão do *sprint*. Apesar da criação deste gráfico ser da responsabilidade da equipa, pode ser o ScrumMaster a atualizá-lo sempre que a equipa não tiver disponibilidade.

Imediatamente antes do término de cada *sprint*, o ScrumMaster organiza uma reunião com a equipa e o dono do produto para que haja uma revisão de todo o trabalho desenvolvido. Esta reunião deve demorar quatro horas para um *sprint* de quatro semanas, e duas horas para um *sprint* de duas semanas.

O objetivo desta reunião é que a equipa Scrum e o dono do projeto discutam o que foi feito e o que não foi feito. Para além disso, a equipa deve mostrar ao dono do projeto o que foi desenvolvido e receber *feedback* relativamente ao mesmo. Por fim, a ideia é perceber se houve alguma mudança ou atualização no âmbito do projeto.

Depois desta reunião e ainda antes do próximo Sprint a equipa deve juntar-se para discutir o que resultou e o que correu mal durante o desenvolvimento daquele *sprint*. A ideia é perceber se é possível tornar a colaboração ainda mais eficiente. Esta reunião deve demorar no máximo três horas para um *sprint* de quatro semanas.

Extreme programming

A seguinte secção é retirada do livro "Extreme programming and agile software development methodologies" da edição de 2004.

Extreme Programming (XP) é uma técnica eficiente, de pouco risco, flexível e preditiva, de desenvolvimento de *software*. É uma disciplina baseada em quatro valores: simplicidade, comunicação, *feedback* e coragem. Foca-se essencialmente numa comunicação clara e na envolvente sólida de trabalho de equipa

Distingue-se das restantes metodologias por:

- Ciclos curtos e precoces com feedback continuo e concreto;
- A sua perspetiva de planeamento incremental, que rapidamente identifica um planeamento holístico do projeto que se espera que evolua ao longo do ciclo de vida do projeto;
- A sua capacidade de agendar as implementações das funcionalidades de maneira flexível, de encontro às necessidades do negócio;
- A sua resiliência na escrita de testes automáticos por parte dos programadores e clientes, para monitorizar o progresso do desenvolvimento no sentido de capturar os erros precocemente;
- Conter um *design* de processo evolutivo que acompanha o projeto até o mesmo terminar;
- Debruça-se nas práticas que se baseiam tanto no trabalho dos instintos de curto prazo dos programadores, como nos interesses de longo prazo dos clientes.

Extreme Programming é uma metodologia que endereça uma grande parte dos riscos identificados no desenvolvimento de *software* e mitiga-os:

- Deslizes no planeamento – Ciclos de lançamento de curto prazo, mitiga o deslize do âmbito do projeto.
- Cancelamento do projeto – XP coloca para produção o entregável mais pequeno e que faça mais sentido para o negócio, para que haja menos erros no lançamento;
- Percentagem de erros – XP cria e mantém um conjunto de casos de teste que são corridos sempre que há alterações. É também suposto o cliente escrever testes por funcionalidade;
- Má interpretação do negócio – XP integra o cliente no projeto. A especificação do projeto é continuamente melhorada ao longo do desenvolvimento.
- Mudanças no negócio – XP tem ciclos de lançamento curtos para que haja menos mudança durante o desenvolvimento de um entregável;
- Funcionalidade errada – XP desenvolve consoante a prioridade das tarefas;
- Mudança da equipa – XP atribui responsabilidade aos colaboradores para apresentarem as estimativas para completar o trabalho e dá-lhes *feedback* sobre o tempo real para melhorar as estimativas. O contacto humano e o incremento nas responsabilidades, são fatores que levam a uma equipa motivada e satisfeita.

Para que haja uma boa implementação da metodologia XP existem um conjunto de práticas que devem ser seguidas, no entanto, nem todas se adequam a todas as situações. Nesse sentido, devem ser adaptadas/escolhidas para cada ocasião as que mais se ajustam. Porém, as práticas trabalham melhor juntas e é mais provável observarem-se melhorias

aquando da implementação conjunta. De qualquer maneira, é possível implementá-las uma a uma e identificar quais as que trazem mais vantagens no desenvolvimento de *software*.

De seguida são apresentadas as práticas para a implementação da metodologia. As mesmas são aplicáveis em qualquer ambiente e produzem uma melhoria rápida no desenvolvimento de *software*:

A equipa sentar-se junta

Desenvolvimento num *open space* suficiente para contemplar toda a equipa. Para ir ao encontro da privacidade da equipa devem existir pequenos espaços ou salas perto do local de trabalho. Este tipo de práticas facilitam a comunicação e interação entre a equipa.

Equipa multidisciplinar

Incluir na equipa competências e perspetivas que sejam necessárias para o sucesso do projeto. A ideia é, tal como o nome da prática indica, ter uma equipa que se complementa com as diferentes competências necessárias. Para além das competências, o sentido de "pertença", "união" e "suporte no trabalho, crescimento e aprendizagem" são também atitudes e valores que devem ser incutidos nas equipas. Se estas forem muito grandes, devem ser subdivididas em equipas mais pequenas para melhorar a resposta às necessidades dos clientes.

Local de trabalho (*workspace*) informativo

"Tornar o local de trabalho sobre o trabalho". Idealmente, um observador aquando a entrada no *workspace,* deveria ter uma ideia geral em pouco tempo de como o projeto está a correr. Esta prática pode ser alcançada colocando autocolantes nas paredes com a informação do projeto. Por exemplo, uma parede com os entregáveis por entregar e outra com o que já foi entregue. Desta maneira é possível perceber o que está a correr bem e o que precisa de ser alterado no planeamento. A imagem seguinte exemplifica uma parede informativa.

Outra implementação informativa no *workspace* são gráficos com o progresso do projeto. Isto é recorrente em projetos cujos requisitos necessitem de um acompanhamento temporal mais próximo. É importante que a informação seja atualizada ao longo do tempo, devendo ser eliminada caso tal não aconteça.

Trabalho energizado

Trabalhar apenas enquanto se é produtivo. Em condições normais é mais fácil identificar erros no desenvolvimento de *software*, enquanto que cansado é difícil reconhecer esse acontecimento.

Programar a pares

Programar a pares consiste em duas pessoas pensarem no mesmo problema ao mesmo tempo. O objetivo é que dois elementos da equipa, programem, analisem, desenhem e testem o sistema na mesma máquina. Dessa maneira existe uma maior clarificação das ideias culminando num sistema mais refinado. Quando aplicada, a programação a pares tem de ter em conta o contexto onde se aplica. A cultura dos colegas de equipa, o espaço pessoal que ele exige, etc, são alguns exemplos que quando escamoteados podem levar a desconforto na equipa.

Utilização do termo histórias

A palavra "requerimentos" é definida como algo obrigatório/mandatário levando a que seja dada um conotação negativa, absolutismo e permanência, inibindo a aceitação da mudança. É nesse sentido que surgem as histórias.

Ciclos semanais

Planear o trabalho semanalmente e reunir no inicio de cada semana. Nesta reunião deve-se rever o progresso até à data, incluindo o progresso da semana passada relativamente ao progresso expectável. Deve-se permitir que o cliente escolha algumas histórias para a implementação da semana seguinte. Por fim, dividem-se as histórias em tarefas permitindo aos membros estimarem e escolherem as mesmas.

***Build* de 10 minutos**

Ter *builds* automatizados são mais valiosos do que aqueles que necessitam de intervenção manual. Para além de mitigar erros, a automatização dos *builds* liberta o *stress* dos programadores.

Integração continua

Integrar e testar as mudanças regularmente. A integração é imprevisível e pode facilmente demorar mais tempo do que o tempo de programação. Quanto maior o tempo entre integrações, maior o custo e a imprevisibilidade.

Elaboração de testes antes de programar

Escrever um conjunto de testes automáticos antes de alterar qualquer código. Efetuar os testes antes de programar resolve os seguintes problemas:

- Obriga a definir explicitamente qual é o objetivo do programa e o que é suposto o mesmo fazer.

- Problema de coesão. Se é difícil efetuar o caso de teste, então é sinal de que pode haver um problema de *design* e não na escrita de teste.
- Confiança. É difícil confiar no autor de um código que não funciona. Ao escrever código funcional e demonstrar as suas intenções com testes automatizados, permite que os seus colegas confiem em si.

Design incremental

Investir no *design* do sistema continuamente. Esforço para manter o *design* do sistema ajustado às necessidades.

Feature driven development

A seguinte secção é retirada do livro "A Pratical Guide to Feature-Driven Development" da edição de 2002.

Feature-driven development é uma metodologia que se foca na produção contínua e tangível de trabalho em todos os níveis do projeto e é baseada no desenvolvimento de *software* iterativo e incremental. Tem uma abordagem direta na construção de sistemas que usam métodos simples de entender e implementar, técnicas de resolução de problemas e relatórios de orientação que fornecem aos *stakeholders* informação para efetuarem decisões em tempo útil. Os programadores recebem informação e a infraestrutura necessária para desenvolver as aplicações. A metodologia permite aos gestores de equipa receberem informação atempada sobre os projetos que possibilita a mitigação dos riscos. A informação do estado atual do projeto permite também tomar decisões controladas e planeadas.

Já os utilizadores, beneficiam com esta metodologia pois conseguem ter áreas do seu negócio automatizadas ao longo do progresso do projeto e têm oportunidade de dar *feedback* construtivo sobre o sistema enquanto o mesmo é desenvolvido.

De seguida apresenta-se as melhores práticas para implementar *feature-driven development*:

Modelar objeto de domínio (*Domain Object Modeling*)

Modelar o objeto de domínio consiste na construção de diagramas de classe identificando os objetos dentro do problema de domínio e a relação entre eles. É uma forma de decomposição de objetos, já que o problema é decomposto em diferentes objetos. O

design e implementação de cada objeto ou classe identificada no modelo é um problema mais pequeno para resolver, já que quando se integram as classes todas é resolvido o problema inicial.

Desenvolvimento por *feature* (característica)

Após a identificação das classes no modelo de objetos de domínio, é possível fazer a implementação de cada uma. O objetivo é restringir os requisitos funcionais aos que criam valor para o utilizador ou cliente, e garantir que os mesmos estão escritos numa linguagem que eles percebam. Estes requisitos funcionais, com estas restrições são também chamados "*features*" (característica). Quando as *features* de um sistema são identificadas, são usadas para conduzir o desenvolvimento nesta metodologia. A entrega de uma peça de infraestrutura pode ser crítica para o projeto, no entanto, não tem qualquer significância para o cliente porque não tem qualquer valor intrínseco para o negócio. Mostrar o progresso em termos de *features* completas, por outro lado, é algo que o cliente consegue perceber e atribuir valor. Os clientes também podem priorizar as *features* em termos de significância para o negócio. O termo *feature* na metodologia FDD é muito especifica, a mesma deve ser pequena e trazer valor para o cliente:

<ação> <resultado> <objeto>

As características devem ser pequenas o suficiente para poderem ser implementadas dentro de duas semanas. Se a função for demasiado complexa e exceda as duas semanas, então deve ser decomposta em várias *features*. Desta maneira, os clientes conseguem medir o progresso frequentemente, melhorando a confiança no projeto, possibilitando *feedback* construtivo numa fase preliminar.

Exemplos de *features:*

- Calcular o total das vendas: Esta *feature* sugere a criação da função calcularTotal() na classe Vendas;
- Aceder à performance do vendedor: Esta *feature* sugere a criação da função acederPerformance() na classe Vendedor.

Responsabilizar as classes

Atribuir classes a pessoas no desenvolvimento dos processos indica quem é, em ultimo caso, o responsável pelo conteúdo de cada classe. As vantagens desta atribuição são as seguintes:

- Ao ser atribuída uma classe a uma pessoa, quando novas mudanças necessitam de ser elaboradas na mesma, a pessoa responsável vai garantir que a classe é mantida e as modificações são devidamente integradas.
- Existe sempre um programador que consegue explicar uma parte do código e como funciona. Isto é especialmente importante para classes muito complexas ou críticas para o negócio.
- O responsável pelo código vai conseguir implementar as novas *features* mais rapidamente do que qualquer outro programador que não está tão familiarizado com aquela parte do negócio.
- O responsável pelo código fica mais motivado pois tem algo feito por ele.

No entanto, este tipo de práticas pode trazer alguns problemas se um programador alterar alguma coisa na sua classe, mas a mesma tiver repercussões ou necessitar de alterações noutras classes. Este tipo de problemas pode levar ao atraso do projeto.

Outra questão está relacionada com a dependência dos programadores. Se algum membro da equipa sair a meio do projeto, ir-se-á perder algum tempo até a equipa perceber como funciona a parte do membro que saiu.

Atribuição de *Features* a equipas

Após terem sido desenvolvidas e identificadas as classes que constituem o modelo, e após a identificação das *features* a serem implementadas e da mesma maneira que se atribuiu classes a membros da equipa, agora tem de se atribuir as *features*. Essa pessoa vai ser responsável por assegurar que a *feature* é desenvolvida corretamente. O desenvolvimento de uma *feature*, provavelmente, vai envolver mais do que uma classe e, nesse caso, mais do que um responsável de classe. Aí, o responsável pela *feature* terá de coordenar uma equipa.

Agora tem-se lideres de um conjunto de *features* e responsáveis por classes e é necessário criar equipas. É difícil garantir que todos os responsáveis de classes fundamentais para desenvolver um *feature* em particular estejam na mesma equipa. Para resolver este problema existem quatro opções:

- Pode-se passar *feature* a *feature* e identificar as classes envolventes e depois separar as mesmas por conjuntos mutualmente exclusivos. Contudo pode não ser possível chegar a esta solução. Se não funcionar passa-se para a opção seguinte.
- Permitir que líderes de equipa peçam a membros de outras para fazerem alterações nas classes que lhes estão alocadas. O problema neste caso é que se tem que esperar que o elemento da outra equipa tenha disponibilidade para desenvolver a tarefa.

- Deixar de existir responsabilidade de classes e passar a haver responsabilidade coletiva.
- Alterar os responsáveis de classes de equipa sempre que esta situação ocorre. Neste caso, os lideres de equipa tem sempre os recursos necessários para desenvolver a *feature.* Não há efetivamente nenhuma regra que impeça os líderes a constituírem equipas dinâmicas que alterem de *feature* para *feature.* Eventualmente os responsáveis por classe podem fazer parte de mais do que uma equipa se necessário.

Inspeções

A metodologia FDD depende muito das inspeções, no sentido de garantir a qualidade do design e do código. O principal propósito das inspeções é a identificação de defeitos. Para além disso, a inspeções têm mais dois benefícios secundários:

1. As inspeções disseminam a cultura e a experiencia do desenvolvimento. Examinar o código de programadores experientes, fazendo com que ele explique as técnicas utilizadas, faz com que os programadores menos experientes aprendam melhores práticas de desenvolvimento;

2. Se os programadores souberem que o seu código não vai passar na inspeção a não ser que esteja em conformidade com os *standarts* das boas práticas, eles mais rapidamente as adotarão.

As inspeções têm de ser feitas de maneira a retirar o medo e o embaraço ou humilhação dos programadores cujo código está a ser inspecionado. Os membros da equipa devem ver as inspeções como uma ferramenta de *debug* e como uma oportunidade de aprender e melhorar com outros programadores.

***Builds* calendarizados regularmente**

O objetivo é em intervalos regulares, pegar no código fonte das *features* desenvolvidas, nas bibliotecas necessárias para a correspondente componente e fazer um *build* do sistema. O período entre *builds* deve-se ajustar ao tamanho do projeto e à duração do mesmo. Desta maneira, são encontrados e corrigidos os erros de integração mais cedo.

Gestão das configurações

A dependência da gestão de configurações depende da complexidade do sistema a ser desenvolvido. A gestão de configurações é, essencialmente, o controlo de versões e manutenção de tudo o que envolve o sistema. Alguns exemplos podem ser, mudança de módulos, requisitos, código fonte, documentação, contratos, etc.

Reportar os resultados

Ter uma visão do estado atual do projeto, saber quão rápido a equipa de desenvolvimento está a adicionar funcionalidades, e ter identificado o resultado final esperado, permite aos líderes e gestores de equipa conduzirem o projeto corretamente com base na informação recolhida.

Dynamic system development method

A seguinte secção é retirada do livro "*Agile Software Development Methods: Review and Analysis*" da edição de 2002.

DSDM é uma metodologia criada em 1994 e que ao longo dos anos a sua aplicação tem crescido principalmente no Reino Unido. A metodologia foca-se essencialmente em fixar o tempo e os recursos do projeto e, posteriormente, ajustar as funcionalidades de acordo com o definido.
Para a correta implementação da DSDM foram criados um conjunto de responsabilidades que devem ser seguidas. Deve incluir a equipa programadores cujo papel é desenvolver a aplicação, um coordenador técnico que define a arquitetura e responsável pela qualidade técnica, um representante de utilizador que tem um papel fundamental ao entregar informação aos programadores referente às necessidades dos utilizadores e aos utilizadores com o progresso do projeto. No sentido de passar a melhor visão possível dos utilizadores deve-se integrar um conselheiro de utilizador que auxilia no fornecimento de informação referente à visão dos utilizadores. Para além destes membros, é também incluído um visionário que tem como papel perceber se o projeto está a ir de encontro ao inicialmente proposto, no fundo, tem a tarefa de assegurar que os requisitos essenciais são descobertos numa fase mais preliminar possível e por fim, o patrocinador executivo que financia o projeto e tem, em último caso, o poder para tomar todas as decisões.
De seguida apresentam-se as boas práticas para a correta implementação do DSDM:

Envolvimento ativo dos utilizadores é imperativo
No sentido de garantir feedback atempado e assertivo é necessário que os utilizadores estejam presentes durante todo o desenvolvimento do sistema.

As equipa têm de ter poder para tomar decisões

A tomada de decisões que sejam longas não é tolerada no ciclo de desenvolvimento rápido. Os utilizadores envolvidos no desenvolvimento têm de ter conhecimento para informar qual o caminho o sistema deve seguir.

O foco é na entrega frequente de produtos

Decisões erradas podem ser corrigidas se o ciclo de entrega for curto e os utilizadores fornecerem *feedback* preciso.

A adequação ao negócio é o critério de aceitação dos entregáveis

Antes de qualquer preocupação relacionada com a técnica do projeto, deve ser dada total importância à satisfação das necessidades do negócio.

É necessário desenvolvimento iterativo e incremental para convergir para a solução do negócio

Os requisitos de sistema raramente ficam os mesmos desde o principio até ao fim do projeto. Ao deixar os sistemas evoluir iterativamente, consegue-se corrigir erros cedo e corrigi-los.

Todas as mudanças durante o desenvolvimento são reversíveis

Durante o desenvolvimento, pode-se adotar o caminho errado. Ao utilizar iterações e garantir que os estados do desenvolvimento podem ser reversíveis, o caminho pode facilmente ser corrigido.

Requisitos são de alto nível

A determinação dos requisitos deve ser apenas de alto nível para permitir a mudança do detalhe dos mesmos se necessário.

Os testes são integradas durante o ciclo de vida do projeto

Com a restrição do tempo os testes tendem a ser negligenciados se deixados para o fim do projeto, nesse sentido, as componentes do desenvolvimento devem ser testadas ao longo do projeto pelos programadores e utilizadores.

Uma colaboração e cooperação dos *stakeholders* é essencial

Para que a metodologia funcione, a organização tem de se envolver e o negócio e o departamento de IT têm de cooperar.

Crystal

A seguinte secção é retirada do artigo (Abrahamsson, Salo, Ronkainen, & Warsta, 2002) de 2002.

A metodologia *Crystal* é formada por um conjunto de metodologias que se adequam a diferentes tipos de projetos e dá especial importância à comunicação e cooperação entre as pessoas. Este conjunto de metodologias não limita as suas práticas de desenvolvimento pondendo, dependendo do contexto, adoptar práticas de XP ou Scrum (ambas metodologias ágeis). *Crystal* caracteriza-se pela entrega regular e incremental de produtos, bem como pela pouca atenção de dá à documentação escrita. O envolvimento direto do utilizador e testes funcionais automatizados também são aspetos intrínsecos na aplicação desta metodologia. *Crystal* também defende a importância de existir mais do que um utilizador a validar cada entrega e que no princípio de cada uma deve existir, se necessário, um ajuste ao projeto.
De seguida apresentam-se as práticas correspondentes à metodologia Crystal:

Planeamento

No início do desenvolvimento de cada entregavél deve ser incluída uma fase de planeamento do novo incremento. Nesta fase a equipa define os requisitos a implementar e entregar na etapa bem como a duração da mesma. Cada incremento deve ter um período de um a quatro meses.

Revisão

Cada incremento está dividido em várias actividades, sendo estas: construção, demonstração e revisão dos objetivos do incremento.

Monitorização

A monitorização da evolução dos incrementos é efetuada da seguinte maneira. São agendadas *milestones* (e.g. revisão 1, revisão 2, testes) que medem o progresso do projeto pela estabilidade que o mesmo apresenta (e.g. estável, não estável).

Estratégia diversificada

O objetivo desta prática é criar equipas pequenas que sejam multifuncionais e com competências em diversas áreas. Desta maneira barreiras como os custos de localização da equipa, comunicação ou coordenação são menores.

Ajuste da metodologia

A metodologia *Crystal* utiliza entrevistas e *workshops* com as equipas no sentido de encontrar qual a metodologia *Crystal* mais adequada para cada projeto. Uma das ideias centrais do desenvolvimento incremental é permitir o aperfeiçoamento das metodologias implementadas com base no conhecimento ganho nas etapas anteriores.

Visualização por parte do utilizador

O entregável deve ser validado por pelo menos dois utilizadores para que seja possível passar à próxima fase.

Pontos de reflexão

Devem ser incluídos períodos de reflexão no início ou fim de cada incremento para que sejam corrigidos eventuais problemas que tenham ocorrido.

Adaptative software development

A seguinte secção é retirada do artigo (Abrahamsson et al., 2002) de 2002.

ASD (Adaptative software development) foca-se, essencialmente, no desenvolvimento de de sistemas grandes e complexos. Este método defende o desenvolvimento incremental e iterativo com testes constantes. O objetivo é fornecer algumas *guidelines* que previnam que este tipo de projetos (grandes e complexos) falhem, sem nunca descorar a importância dada à criatividade da equipa. ASD é orientado às componentes e não ás tarefas, isto é, foca-se mais nos resultados e na sua qualidade do que nos processos usados para produzir os mesmos.

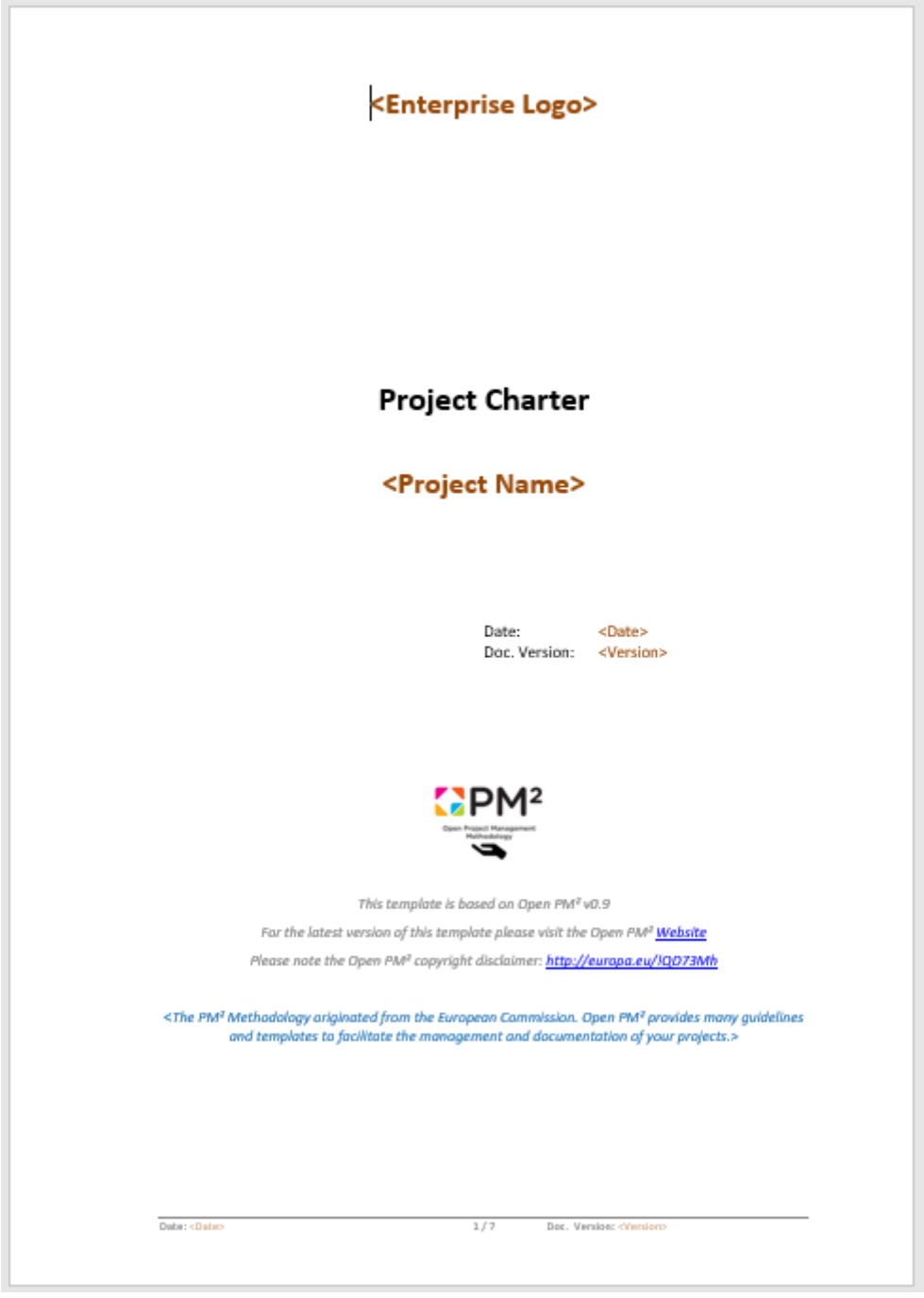
Relativamente às práticas, esta metodologia propõe apenas três, sendo estas: 1ª – Desenvolvimento iterativo; 2ª – planeamento por componentes; 3ª – Foco no cliente (reuniões). A dificuldade em identificar as suas práticas é considerada o maior problema desta metodologia.

Apêndices

Apêndice A

Requirements List									
ID	**Sprint ID**	**Deliverable ID**	**Category**	**Title**	**Description**	**Acceptance Criteria**	**Effort**	**Priority**	**Delivery Date**
Guidelines			*<Choose the requirement category, technical requirement, project requirement, functional requirement, non functional requirement>*	*<Short title for the requested change>*	*<Description of the recommended action(s), including steps, deliverables, timescale and resources.>*	*<Small description concerned the acceptance criteria. What does this requirement need to have to be considered complete(non obligatory)*	*<A numerical value denoting how much effort will be necessary to implement the change: 5- Very High to 1- Very Low>*	*<A numerical value denoting the agreed priority of the change: 5- Very High to 1- Very Low>*	*<Target date for the change to be delivered: <dd/mm/yy>>*
C01									
C02									
C03									
C04									
C05									
C06									
C07									
C08									
C09									
C10									
C11									
C12									
C13									
C14									
C15									
C16									
C17									
C18									
C19									
C20									

Apêndice B



<Enterprise Logo>

Project Charter

<Project Name>

Date: <Date>
Doc. Version: <Version>

PM²
Open Project Management Methodology

This template is based on Open PM² v0.9

For the latest version of this template please visit the Open PM² Website

Please note the Open PM² copyright disclaimer: http://europa.eu/!QD73Mh

<The PM² Methodology originated from the European Commission. Open PM² provides many guidelines and templates to facilitate the management and documentation of your projects.>

Date: <Date> 1 / 7 Doc. Version: <Version>

Document Control Information

Settings	Value
Document Title:	Project Charter
Project Title:	<Project Name>
Document Author:	<Document Author>
Project Owner:	<Project Owner (PO)>
Solution Provider	<Solution Provider (SP)>
Project Manager:	<Project Manager (PM)>
Doc. Version:	<Version>
Sensitivity:	<Public, Limited, High>
Date:	<Date>

Document Approver(s) and Reviewer(s):

NOTE: All Approvers are required. Records of each approver must be maintained.

All reviewers in the list are considered required unless explicitly listed as Optional.

Name	Role	Action	Date
		<Approve / Review>	

Document history:

The Document Author is authorized to make the following types of changes to the document without requiring that the document be re-approved:

- Editorial, formatting, and spelling
- Clarification

To request a change to this document, contact the Document Author or Project Owner.

Changes to this document are summarized in the following table in reverse chronological order (latest version first).

Revision	Date	Created by	Short Description of Changes

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

<This section should provide an executive summary. Complete this section last.>

1 PROJECT DESCRIPTION

1.1 Scope

*<This section should identify what it is considered as in scope of the project, i.e., the outputs that the project **WILL** ~~deliver~~ and which form the solution which addresses the current situation (problem, need or opportunity). The definition of the scope can be complemented with the scope of organisational change management activities associated with the implementation of the project and required to achieve the intended benefits.>*

1.2 Success Criteria

<This section should describe the success criteria of the project. Think of success criteria as the measurable criteria based on which the project as a whole can be deemed as a success or a failure.>

<Critical criteria for the project are those which in their absence the project cannot be considered a success. Success criteria may be on scope, schedule and costs. Try to distinguish any product success criteria from the overall project success criteria, in a way that the latter can relate to the project's expected outcomes.>

<Example: Training project –"minimum of 4500 staff members trained in 19 Member States from the project target of 5000 staff members in 20 countries".>

1.3 Stakeholder and User Needs

<This section should list the key needs of the stakeholders and users that the project shall address.
(A user is considered as a group – or individual – that will use one or more of the project's outputs).

Use the questions below to help you describe each need:

- *Who is the stakeholder of this need?*
- *What is the need? What solutions does the stakeholder want?*
- *What are the reasons that justify addressing this need?*
- *How is it currently addressed?*

It's also important to indicate the relative importance of each need (from the stakeholder/user perspective). Ranking and cumulative voting techniques can reveal needs that must be addressed versus needs that stakeholders/users would like to be addressed (potentially).>

	Need Description	Priority

1.4 Deliverables

<This section should identify the deliverables or outputs of the project. Think of deliverables as a tangible or intangible object produced as a result of the project that is intended to be delivered to the Project Owner's organisation. A deliverable could be an automated report, a document, a server, a new policy or regulation, a conference, a training, a campaign, etc. A deliverable may be composed of multiple sub-deliverables.

Note that the standard project management deliverables (e.g. the PM² Artefacts) should not be considered in this section.>

	Deliverable Name	Deliverable Description

1.5 Requirements

<This section should define the expected features of the outputs that will be delivered to the users and to the project owner organisation. Think of requirements as the high-level capabilities associated to the outputs that are necessary to deliver the expected benefits to the users. At this moment of the project, keep requirement descriptions at a high level and focus on capabilities needed and why (not how) they should be implemented. These requirements will be expanded in greater detail later in the project as it becomes necessary to detail how these requirements will be implemented by the project core team.

To structure the way the requirements are identified and described, align them with the previously defined stakeholders and user needs. Keep in mind that one need may be answered by implementing several requirements.

Because this document is reviewed and read by a wide variety of stakeholders, the level of detail should remain general enough for everyone to understand it. However, enough detail must be available to provide the next stages of the project with the information needed to detail how the outputs will respond to the stakeholders and user needs.

Note: An example of a feature for an issue tracking system (IT System) might be the ability to provide a specific type of report. As the use-case model takes shape, it is recommended that you update their descriptions to refer to the use cases that detail them.>

Related Need	Requirements	Deliverable(s)
<Please refer to the identifier of the need (ID).>		

<Example:

- *Need: consumers face problems in selecting the best and cheapest energy company that provides electricity and gas for homes.*
- *Deliverable: a website to inform customers with up-to-date information on energy providers, prices, and terms & conditions.*
- *Feature: website should allow the consumer to enter the most relevant selection criteria to select the best fitting energy provider in the market.*

Note: Features and needs are examples of Requirements. For advanced Requirement Management, you can refer to the PM² Requirement Management Plan and/or use the Requirement Traceability Matrix artefact.>

1.6 Constraints

<This section should describe any project constraints that affect the way we can manage this project. Constraints can come from areas such as the people that can be involved, schedule, budget, resources, products to be reused or acquired, technology to be employed and interfaces to other products. List the project constraints based on the current knowledge.

Also list decisions and compliance related constraints. Mention constraints that arise both from the organisation as well as from the external (to the project or/and organisation) environments.

If a separate document does not exist, you can include information related to security constraints, document management constraints, data protection constraints, or other.>

1.7 Assumptions

<This section should describe any project assumptions related to business, technology, resources, organisational environment, scope, expectations, or schedules.

At this stage, assumptions are considered to be facts (true); ~~however~~ they need to be further validated to ensure that they are indeed facts. Note that assumptions that have not been validated may became risks.>

1.8 Risks

<This section should highlight the key project risks that are identified at this initial stage and proposes corresponding risk management strategies. This initial risk assessment does not replace the full risk assessment that is conducted during the planning phase. You may refer to the project's Risk Log for a complete list and description of risks and corresponding actions – provide a link to the project's Risk Log.>

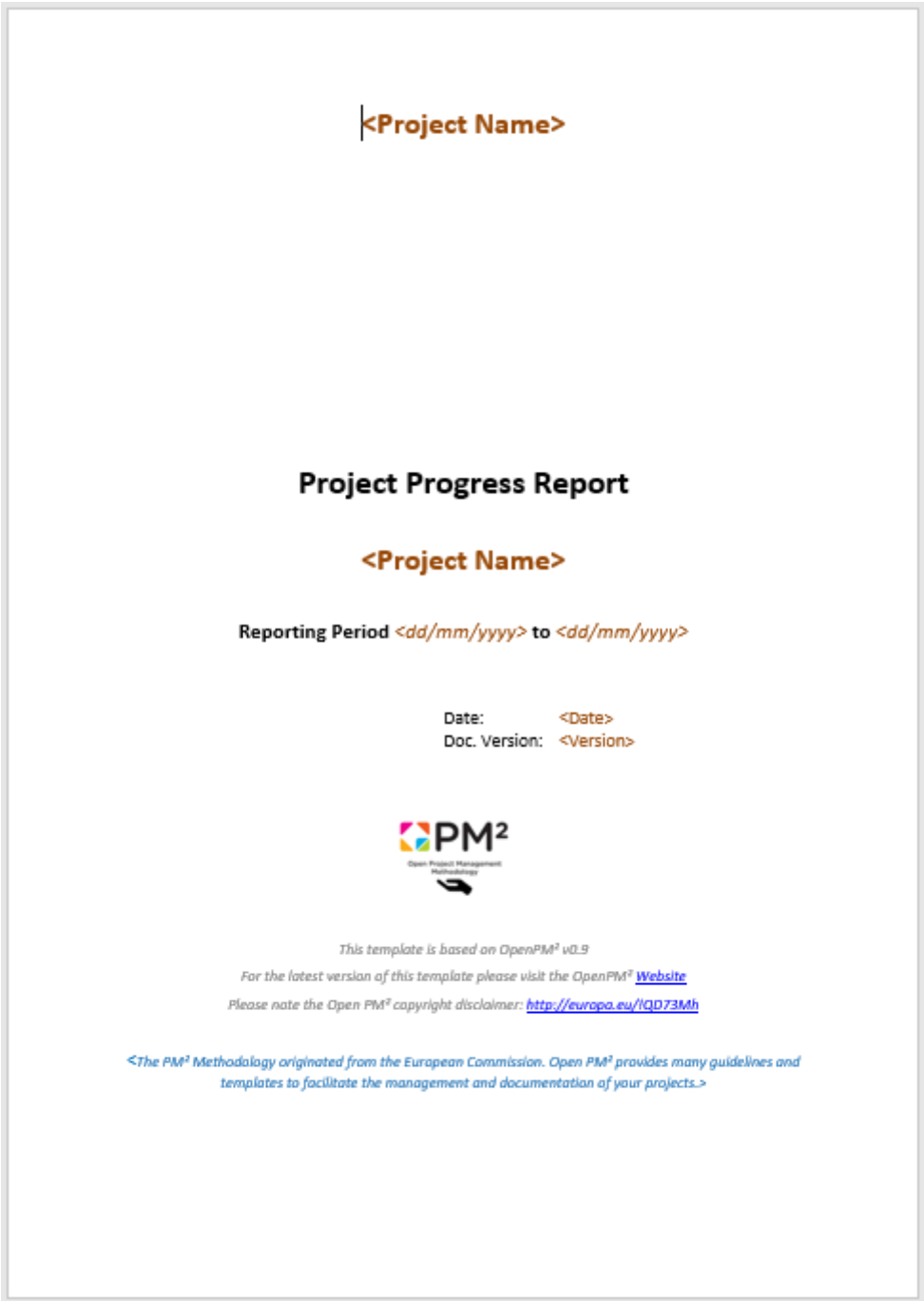
ID	Risk Description & Details	Status	Likelihood	Impact	Risk Owner	Risk Response Strategy

2 Roles and Responsibilities

*<This section should describe the Roles and Responsibilities of the Project Governance. You can refer to the PM² Methodology and include the assignment of the Standard PM² **Roles and Responsibilities**, or/and define any deviations from the Standard.>*

Name	Position	Role
Name of stakeholder.	*Position in the organization.*	*Role in the project*

Apêndice C



<Project Name>

Project Progress Report

<Project Name>

Reporting Period <dd/mm/yyyy> to <dd/mm/yyyy>

Date: <Date>
Doc. Version: <Version>

PM²
Open Project Management Methodology

This template is based on OpenPM² v0.9

For the latest version of this template please visit the OpenPM² Website

Please note the Open PM² copyright disclaimer: http://europa.eu/!QD73Mh

<The PM² Methodology originated from the European Commission. Open PM² provides many guidelines and templates to facilitate the management and documentation of your projects.>

Document Control Information

Settings	Value
Document Title:	Project Progress Report
Project Title:	<Project Name>
Document Author:	<Document Author>
Project Owner:	<Project Owner (PO)>
Project Manager:	<Project Manager (PM)>
Doc. Version:	<Version>
Sensitivity:	<Public, Limited, High>
Date:	<Date>

Document Approver(s) and Reviewer(s):

NOTE: All Approvers are required. Records of each approver must be maintained. All Reviewers in the list are considered required unless explicitly listed as Optional.

Name	Role	Action	Date
		<Approve / Review>	

Document history:

The Document Author is authorized to make the following types of changes to the document without requiring that the document be re-approved:

- Editorial, formatting, and spelling
- Clarification

To request a change to this document, contact the Document Author or Owner.
Changes to this document are summarized in the following table in reverse chronological order (latest version first).

Revision	Date	Created by	Short Description of Changes

TABLE OF CONTENTS

1. PROJECT OVERVIEW

1.1. Report the state of project

< The section should provide a high-level overview of the entire project and the actual status. May include the following elements such as: overall outcomes and business triggers, overall description of the solution, major changes in scope, resources, constraints, achievements, etc.>

1.2. Deliverables

<In case of the yearly reporting, this section should address the full lifespan of the project and should not focus exclusively on the reporting period. The objective is to provide an overview for the complete project duration.>

ID	Deliverable Name	Target Delivery Date	Actual Delivery date	Status	Comments
				<on-going, planned, achieved>	

<The deliverable IDs should be aligned with the ones used previously in the Project Plan.>

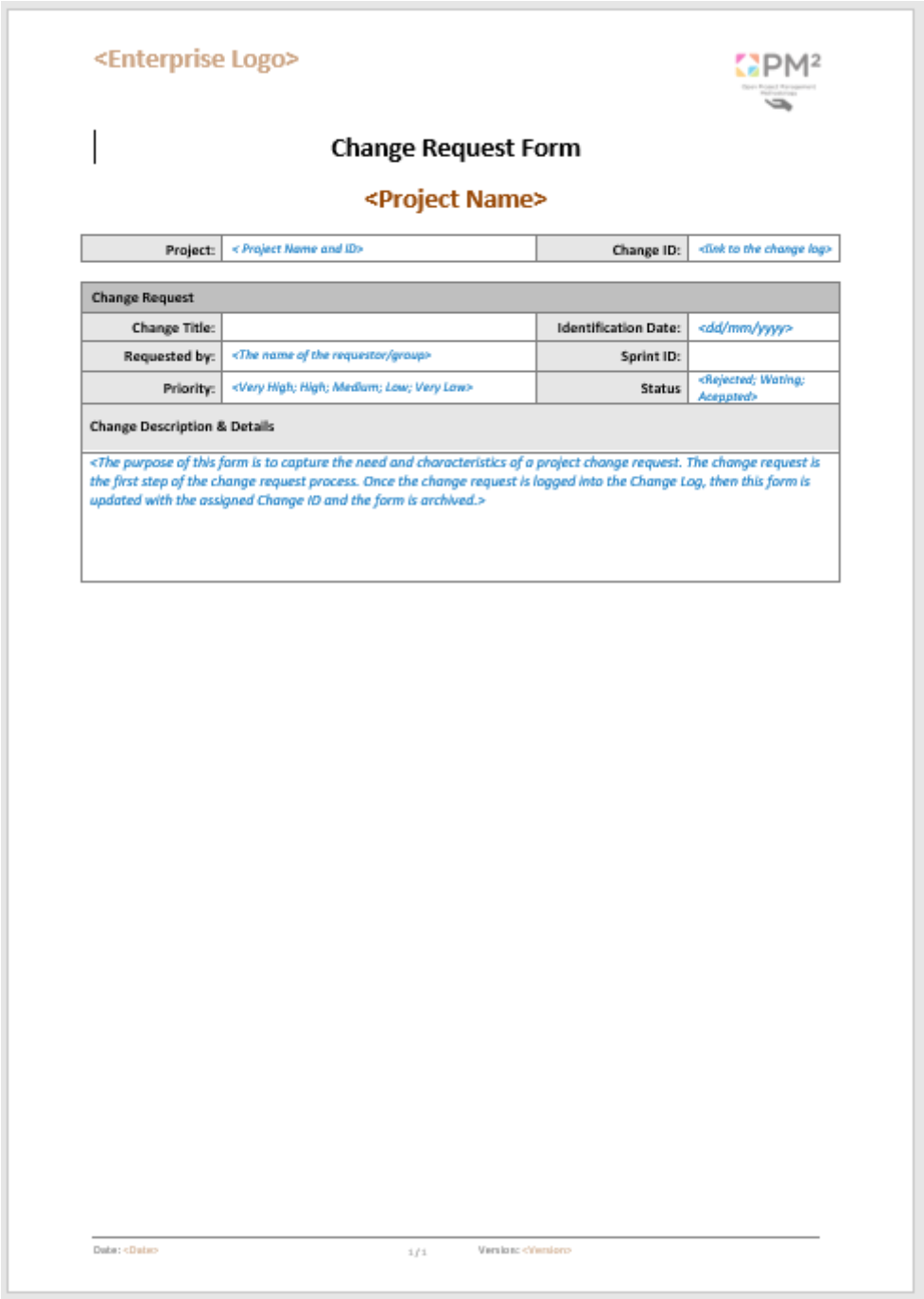
1.3. Requirements

ID	Requirement	Target Delivery Date	Actual Delivery date	Status	Comments
				<on-going, planned, achieved>	

1.4. Remarks and Comments

<In this section you should and some comments that are useful for the rest of the project>

Apêndice D



<Enterprise Logo>

PM²

Change Request Form

<Project Name>

Project:	< Project Name and ID>	Change ID:	<link to the change log>

Change Request			
Change Title:		Identification Date:	<dd/mm/yyyy>
Requested by:	<The name of the requestor/group>	Sprint ID:	
Priority:	<Very High; High; Medium; Low; Very Low>	Status	<Rejected; Wating; Accepted>
Change Description & Details			
<The purpose of this form is to capture the need and characteristics of a project change request. The change request is the first step of the change request process. Once the change request is logged into the Change Log, then this form is updated with the assigned Change ID and the form is archived.>			

Date: <Date> 1 / 1 Version: <Version>

Apêndice E

<Enterprise Logo>

Project Plan

<Project Name>

Date: <Date>
Doc. Version: <Version>

PM²
Open Project Management Methodology

This template is based on Open PM² v0.9

For the latest version of this template please visit the Open PM² Website

Please note the Open PM² copyright disclaimer: http://europa.eu/!QD73Mh

<The PM² Methodology originated from the European Commission. Open PM² provides many guidelines and templates to facilitate the management and documentation of your projects.>

Document Control Information

Settings	Value
Document Title:	Project Plan
Project Title:	<Project Name>
Document Author:	<Document Author>
Internal Project Manager:	<Project Manager (IPM)>
External Project Manager:	<Project Manager (EPM)>
External Team	<External Team (ET)>
Project Type:	<Internal development; External development>
Doc. Version:	<Version>
Date:	<Date>

Document Approver(s) and Reviewer(s):

NOTE: All Approvers are required. Records of each approver must be maintained.
All reviewers in the list are considered required unless explicitly listed as Optional.

Name	Role	Action	Date
		<Approve / Review>	

Document history:

The Document Author is authorized to make the following types of changes to the document without requiring that the document be re-approved:

- Editorial, formatting, and spelling
- Clarification

To request a change to this document, contact the Document Author or Project Owner.
Changes to this document are summarized in the following table in reverse chronological order (latest version first).

Revision	Date	Created by	Short Description of Changes

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

<This section should provide an executive summary. Complete this section last.>

2 PROJECT DESCRIPTION

2.1 Scope

*<This section should identify what it is considered as in scope of the project, i.e., the outputs that the project **will** ~~deliver~~ and which form the solution which addresses the current situation (problem, need or opportunity). The definition of the scope can be complemented with the scope of organisational change management activities associated with the implementation of the project and required to achieve the intended benefits.>*

2.2 Assumptions

<This section should describe any project assumptions related to business, technology, resources, organisational environment, scope, expectations, or schedules.

At this stage, assumptions are considered to be facts (true); ~~however~~ they need to be further validated to ensure that they are indeed facts. Note that assumptions that have not been validated may become risks.>

2.3 Roles and Responsibilities

*<This section should describe the Roles and Responsibilities of the Project Governance. You can refer to the PM³ Methodology and include the assignment of the Standard PM³ **Roles and Responsibilities**, or/and define any deviations from the Standard.*

Delete the table that doesn't match you project type (internal development; external development).>

Development team (EQ) Team composed by programmers that develop the project functionalities;

Internal project manager (GIP) IGFEJ manager that controls the project internally;

External project manager (GEP) External manager that controls the development team;

Infrastructure ~~Responsable~~ (RI) IGFEJ collaborator who has the authority to make decisions regarding the infrastructures necessary for the elaboration of the project;

IGFEJ director (DI) A person should be appointed to represent IGFEJ and be responsible for the project. This person should be a Vowel or a department director;

Justice Ministry (MJ) In this scenario is the client that requests the service to IGFEJ.

Intern Project

RACI	DI	MJ	GPI	EQ	RI
Phases					
Requirements gathering	I	A	R		
Sprint planning	I	I/C	A/R	C	I
Sprint execution		C	A	R	R
Sprint revision	I	A	R	R	R

Project evaluation	I	A	R	I	
Sprint replanting	I	I/C	A/R	C	C
Project summary	I		A/R	C	C
Documents					
Requirements list		A	R		
Progress Report	I	I	A	R	
Project plan	I	I/C	A/R	C/I	I
Project charter	I	R/A	R/I/C		
Change requirement		R/A	R/A	C/I	

Turnkey project

RACI	DI	MJ	GPI	GPE	EQ	RI
Phases						
Requirements gathering	I	A	R			
Sprint planning	I	I/C	A/C	R	C	C
Sprint execution		C	A	R	R	R
Sprint revision	I	A	R	R	R	R/C
Project evaluation	I	A	R			
Sprint replanting	I	I/C	A	R	C/I	C
Project summary	I		A/C	R	C	C
Documents						
Requirements list		A	R			
Progress Report	I	I	A	R	C/I	
Project plan	I	I/C	A	R	I	C
Project charter	I	R/A	R/I/C			
Change requirement		R/A	R/A	C/I		

A .....: Accountable

R .....: Responsible

C .....: Consulted

I .....: Informed

2.4 Project Dependencies or Interrelations

<Identify any dependencies or interrelations of this project with other work or projects that has/is/will be undertaken, or with other problems or solutions. For example, the project may be part of a programme or a network of projects each contributing towards a common goal.

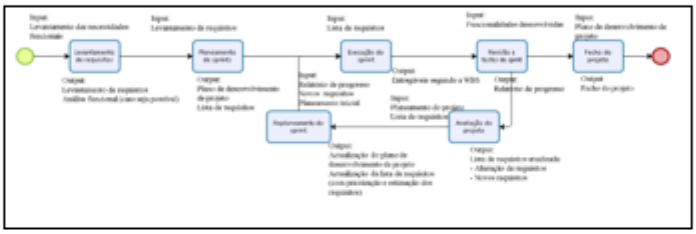
Identifying and documenting these dependencies can influence the project management priorities, the management approach, can results additional objectives, or simply result in constraints and/or risks.>

2.5 Project Lifecycle

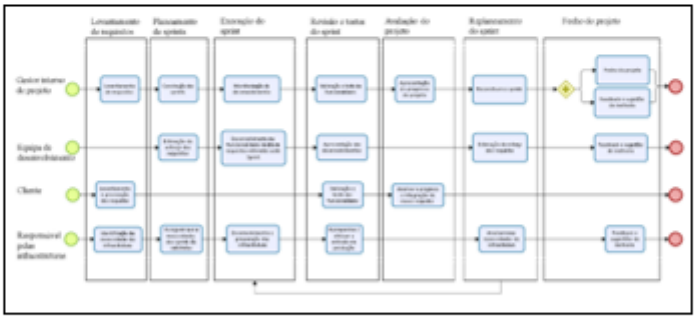
<Present the project management lifecycle (phases) to be used in the project and how the project will move forward from one phase to the next (i.e. the planned approvals or gates), and describe any deviations from the standard

PM² project management lifecycle.

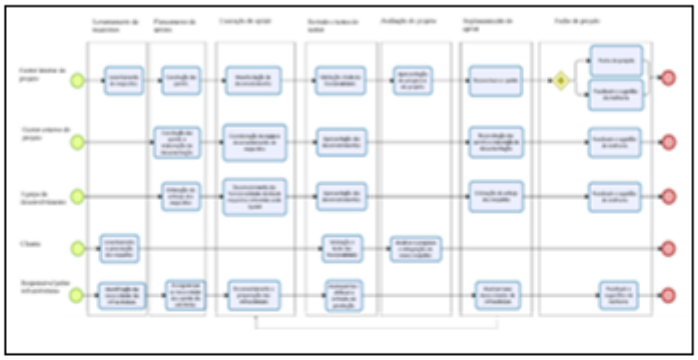
Delete the table that doesn't match you project type (internal development; external development).>



Internal project



External project



2.6 Sprints

<This section should identify the deliverables or outputs of the project. Think of deliverables as a tangible or intangible object produced as a result of the project that is intended to be delivered to the Project Owner's organization. A deliverable could be an automated report, a document, a server, a new policy or regulation, a conference, a training, a campaign, etc. A deliverable may be composed of multiple sub-deliverables.

Note that the standard project management deliverables (e.g. the PM² Artefacts) should not be considered in this section.>

Sprint ID	Deliverable ID	Deliverable Title	Deliverable Description	Delivery date

2.7 Work Breakdown

This section presents the breakdown of the project into smaller and more manageable components such as deliverables, work packages, activities, and tasks. Each lower level of the representation offers a finer level of detail of the deliverables and work that all together define the project output(s) and the work involved to produce them.

Work Breakdown			Description	Acceptance criteria	Start Date	End Date
1.0		**Project**	-			
1.1		**Sprint 1**				
1.1.1		Deliverable				
1.1.1.1		Task				
1.1.1.2		Task				
1.1.2		Deliverable				
1.1.2.1		Task				

1.1.2.2		Task				
1.1.3		Deliverable				
1.1.3.1		Task				
1.1.3.2		Task				
1.2		**Sprint 2**				
1.2.1		Deliverable				
1.2.1.1		Task				
1.2.1.2		Task				
1.2.2		Deliverable				
1.2.2.1		Task				
1.2.2.2		Task				
1.2.3		Deliverable				
1.3		**Sprint 3**				
1.3.1		Deliverable				
1.3.1.1		Task				
1.3.2		Deliverable				
1.3.3		Deliverable				