Universidade do Minho
Departamento de Sistemas de Informação

Juliana Sofia Cardoso Soares

AgileMIP: método ágil de implantação de TI

Dissertação de Mestrado
Mestrado em Sistemas de Informação

Desenvolvido sob orientação do
Professor Doutor João Eduardo Quintela Varajão
Professor Doutor João Álvaro Carvalho

junho de 2020

DECLARAÇÃO DE INTEGRIDADE

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

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

Agradecimentos

Há dois anos atrás, comuniquei a minha decisão de ingressar no Mestrado em Sistemas de Informação (MSI). Comecei por um professor, que me respondeu: "é difícil, mas com trabalho, tu consegues". Foi com este *mindset* que ingressei no MSI, e aqui estou. Este trabalho de dissertação marca o final de um capítulo, e representa um conjunto de emoções: evolução, aprendizagem, melhoria, perseverança, e, principalmente, conquista.

Aqui, gostaria de agradecer a todos os que, de alguma forma, influenciaram o meu percurso académico e tiveram um contributo na concretização deste marco.

Em primeiro lugar, gostaria de agradecer à minha família, que fui construindo ao longo do tempo: Miguel, avó, padrinho, mãe, "mana" Daniela, Fernando, Sandra, Diana, Jota, Ju, Tiago, Diogo e Cidália. Obrigada por acreditarem em mim e, talvez sem perceberem, me incentivarem e ajudarem a ultrapassar todos os obstáculos. Em especial, ao Miguel, por estar sempre lá, incentivar-me a ser cada vez melhor, e por acreditar incondicionalmente nas minhas capacidades e ideias.

De seguida, gostaria de agradecer a todos os professores que influenciaram o meu percurso académico, especialmente, aos orientadores desta dissertação, professor João Varajão e João Álvaro Carvalho. Ao Professor Varajão, pelo constante acompanhamento e *feedback*, pela sua disponibilidade, prontidão, compreensão, e pelo encorajamento para fazer sempre mais e melhor. Ao professor João Álvaro Carvalho, por todo o seu *feedback*, pela partilha de conhecimento e pela sua disponibilidade.

Gostaria também de agradecer à Primavera BSS, sem a qual esta dissertação não teria lugar. Em particular, agradeço à Primavera *Academy*, à CSU, e, em especial, à Carina, por todos os incentivos e ideias, e pela preocupação em demonstrar o tom cinza, ao invés de apenas o preto e o branco.

A todos,
Um grande obrigada.

May the force be with you.

Resumo

O Desenvolvimento de Sistemas de Informação (DSI) é um processo complexo, composto por duas dimensões (organizacional e tecnológica), cujo objetivo é melhorar as organizações. Para a sua realização, podem ser contemplados diferentes ciclos de vida: em *waterfall*, *agile*, iterativo, incremental ou híbrido, os quais se encontram refletidos em métodos, modelos, processos, *frameworks*, técnicas, etc.

O termo "*agile*" encontra-se tipicamente associado ao desenvolvimento de *software*. Todavia, na atualidade, a sua aplicabilidade é muito mais abrangente, podendo ser perspetivado como um *mindset* e sendo amplamente utilizado noutras áreas, como é o caso da gestão de projetos.

Na atualidade, a Primavera BSS utiliza a Metodologia de Implementação Primavera (MIP) para implantar os seus produtos, que traduz um ciclo de vida tipicamente em *waterfall*. No contexto do seu processo de inovação tecnológica, a Primavera BSS irá em breve lançar um novo ERP nativo em *cloud*, o que impulsionou a revisão da MIP.

A finalidade desta dissertação consiste na proposta de um novo método de implantação que incorpore o *mindset* ágil e que traga agilidade à implantação dos produtos da Primavera BSS. Entre outros, o trabalho incluiu uma revisão da literatura sobre o DSI, sobre os diferentes ciclos de vida existentes, uma análise comparativa dos modelos de desenvolvimento de *software*, a análise do *agile* sob diferentes perspetivas, a exploração do processo de implantação e métodos de implantação existentes no mercado, uma análise comparativa dos métodos ágeis de implantação identificados, a proposta de um novo método ágil de implantação, e a sua avaliação.

Para tal, foram seguidas as orientações do *Design Science Research*, contemplando seis etapas: definição do problema, definição dos objetivos da solução, *design* e desenvolvimento, demonstração, avaliação e comunicação.

Espera-se que este novo método de implantação permita um melhor desenvolvimento dos projetos de implantação da Primavera BSS, superando a dificuldade sentida no fecho dos projetos e respondendo à necessidade de dispor de um método ágil de implantação dos produtos Primavera, potenciando maior agilidade, maior satisfação do cliente, e o maior sucesso dos projetos.

Palavras-chave: *agile*, desenvolvimento de sistemas de informação, implantação, implementação, *method engineering*, métodos ágeis, métodos de implantação, *mindset* ágil, modelo em espiral, modelo em *waterfall*, modelo incremental, *unified process*,

Abstract

Information Systems Development (ISD) is a complex process, composed of two dimensions (technological and organizational), whose goal is to improve organizations. For its realization, different life cycles can be contemplated: waterfall, agile, iterative, incremental, and hybrid, which are reflected in methods, models, processes, *framework*s, techniques, etc.

The term "agile" is typically associated with software development. However, nowadays, its applicability is much broader, being able to be seen as a *mindset*, and widely used in other areas, such as project management.

Currently, Primavera BSS uses the Metodologia de Implementação Primavera (MIP) to implant its products, which corresponds to a typical waterfall life cycle. In the context of its technological innovation process, Primavera BSS will soon launch a new ERP cloud-native, that boosted the MIP review.

The purpose of this dissertation is to propose a new agile implantation method, that incorporates the agile *mindset* and brings agility to the implantation process of Primavera BSS. Among others, the work included a literature review on the ISD, on the different existing life cycles, a comparative analysis of software development models, an analysis of agile from different perspectives, the exploration of the implantation process and implantation methods on the market, a comparative analysis of the identified agile implantation methods, the proposal of a new agile implantation method, and its evaluation.

To this end, the orientations of Design Science Research were followed, contemplating six steps: definition of the problem, the definition of the objectives of the solution, design and development, demonstration, evaluation, and communication.

It is expected that this new implantation method will allow a better development of Primavera BSS implantation projects, overcoming the difficulty felt in closing projects and responding to the need for an agile method for the implantation of Primavera products, enhancing greater agility, greater customer satisfaction, and greater projects success.

Keywords: agile, agile methods, agile mindset, implantation, implantation methods, implementation, incremental model, information systems development, method engineering, spiral model, unified process, waterfall model

Índice

Índice de figuras

Índice de tabelas

Abreviaturas, Siglas e Acrónimos

DSI	Desenvolvimento de Sistemas de Informação
DSR	*Design Science Research*
ERP	Enterprise Resource Planning
ME	*Method Engineering*
MIP	Metodologia de Implementação Primavera
Primavera BSS	Primavera *Business Software Solutions*
SI	Sistema de Informação
TI	Tecnologias da Informação

1 Introdução

Neste capítulo é apresentado o âmbito da presente dissertação. Em primeiro lugar, é realizado o seu enquadramento (1.1), seguindo-se a apresentação da motivação para a sua realização (1.2), da sua finalidade (1.3), e da estrutura do documento (1.4).

1.1. Enquadramento

Os Sistemas de Informação (SI) e o Desenvolvimento de Sistemas de Informação (DSI) são interpretados e analisados de diferentes perspetivas por diferentes autores (Korpela et al., 2002; Carvalho, 2000).

No âmbito desta dissertação, o DSI é definido como um processo através do qual uma atividade coletiva é facilitada por uma nova Tecnologia da Informação (TI), através de atividades como análise, *design*, implantação, introdução e suporte (Korpela et al., 2002), composto por duas dimensões: organizacional e tecnológica (Xia e Lee, 2004; Benbya e McKelvey, 2006; Carugati, 2008).

Para o DSI, existem diversos ciclos de vida, que se encontram refletidos em métodos, modelos, *frameworks,* etc. O *Project Management Institute* e a *Agile Alliance* (PMI, 2017; Agile Alliance e PMI, 2017) identificam cinco tipos de ciclos de vida: em *waterfall* (ou preditivo ou orientado ao planeamento), incremental, iterativo, *agile* (ou adaptativo) e híbrido.

O ciclo de vida em *waterfall* segue uma ordem sequencial e tem associada alguma rigidez e inflexibilidade relativamente à mudança. Por sua vez, os ciclos de vida ágeis, que têm ganho atenção nos últimos anos, são focados nas pessoas do projeto, na mudança, adaptação, na entrega e atualização frequente, e na constante priorização do trabalho (Agile Alliance e PMI, 2017). Os ciclos iterativos e incrementais situam-se entre estes dois ciclos de vida, providenciando produtos funcionais, enquanto mantêm o controlo das variáveis do projeto e das suas mudanças (PMI, 2017).

Os valores e princípios do *agile* foram definidos em 2001 com o *Agile Manifesto* (Beck et al., 2001), e podem ser sintetizados através de quatro termos: colaboração, entrega, reflexão e melhoria (Cockburn, 2016). Estas caraterísticas, bem como as competências da equipa e o envolvimento do cliente, são elementos chave no sucesso de projetos ágeis (Tam et al., 2020), sendo o capital humano e as competências comportamentais um fator de sucesso destes projetos (Varajão, 2019).

O *agile* introduziu um elemento crucial, a agilidade, que consiste na capacidade de uma organização rapidamente lidar, aceitar e adaptar-se à mudança e às situações do ambiente (Bishop et al., 2018). Na atualidade, já não é apenas relacionado com as TI, incluindo situações complexas, de diferentes dimensões, e diferentes áreas do conhecimento (Axelos, 2018). Assim, pode ser visto como um *mindset*, caraterizado por um conjunto de valores e princípios, como a capacidade de resposta à mudança, maior

visibilidade sobre o projeto, flexibilidade, maior envolvimento do cliente, entrega rápida de funcionalidade, e maior colaboração e trabalho em equipa (Deloitte, 2020).

Áreas como a gestão de projetos já avaliaram a aplicabilidade do *agile* enquanto *mindset*, sendo já amplamente utilizado em diferentes tipos de projeto (Gustavsson, 2016). No entanto, no contexto de implantação de TI, o *agile* ainda não é amplamente explorado (Isetta e Sampietro, 2018), tornando-se importante investigar sobre a sua aplicabilidade neste contexto.

1.2. Motivação do projeto

A Primavera BSS é uma *software house* portuguesa que se dedica ao planeamento, desenvolvimento e comercialização de aplicações de gestão, que se mostram adequadas a qualquer negócio, independentemente da sua dimensão (Primavera BSS, 2020).

As soluções Primavera são numerosas e consideravelmente variadas, no entanto, podem ser organizadas em quatro categorias: serviços *cloud e mobile* (plataformas de desenvolvimento, faturação eletrónica, pagamentos eletrónicos e menú digital); soluções setoriais (administração pública, construção, consultoria e contabilidade, restauração e retalho); ferramentas de *reporting* (*business analytics, office extension e fiscal reporting*); e soluções especializadas (gestão da manutenção, vendas em mobilidade, *employee e manager self-service,* e distribuição e logística).

Para a implantação dos seus produtos, a Primavera BSS possui uma metodologia própria, a MIP - Metodologia de Implementação Primavera, com um ciclo de vida tipicamente em *waterfall*. No entanto, no âmbito do seu processo de inovação tecnológica, encontra-se a desenvolver um novo ERP nativo em *cloud*, denominado Rose, que impulsionou a revisão da metodologia de implantação dos produtos Primavera.

Assim, surge o presente trabalho de dissertação, cujo objetivo é propor um novo método de implantação de TI (designado AgileMIP) para as aplicações *cloud* e *on-premise*, que procura trazer agilidade ao contexto de implantação da Primavera BSS e dos seus parceiros, suportar os processos e os métodos de trabalho inerentes às equipas Primavera, e superar as dificuldades e limitações atualmente sentidas com a MIP.

1.3. Objetivo do trabalho e síntese do plano de investigação

Esta dissertação tem como objetivo propor um novo método ágil de implantação de Tecnologias da Informação (TI). O seu objetivo é introduzir agilidade na implantação dos produtos Primavera, suportar os processos de negócio e trabalho da Primavera BSS, e superar a dificuldade atualmente verificada no fecho dos projetos.

Entre outras atividades, este trabalho inclui a revisão da literatura sobre o Desenvolvimento de Sistemas de Informação (DSI) e os seus diferentes ciclos de vida, a identificação e análise de modelos de desenvolvimento de *software*, a análise das diferentes dimensões do *agile* (no desenvolvimento de *software*, na gestão de projetos, enquanto *mindset*, e na implantação de TI), a análise do processo de implantação, e, também, o estudo do processo de criação de métodos. O objetivo principal é propor de um novo método de implantação de TI.

Com a elaboração desta dissertação, pretende-se responder às seguintes questões de investigação:

- Questão 1: Quais as caraterísticas do *mindset* ágil no contexto de implantação de TI?
- Questão 2: Que caraterísticas deve reunir um método ágil para a implantação dos produtos Primavera?

Para o desenvolvimento desta dissertação foram utilizadas as orientações do *Design Science Research (DSR)*, uma vez que se revela adequada para um trabalho que visa desenvolver um método. Nesta dissertação foram seguidas as etapas de DSR sugeridas por Peffers et al. (2007): identificação do problema e motivação, definição dos objetivos da solução, conceção e desenvolvimento, demonstração, avaliação, e, por fim, comunicação.

1.4. Estrutura do documento

Nesta secção é apresentada, sucintamente, a organização e conteúdo deste documento, o qual é composto por seis capítulos.

O Capítulo 1 corresponde à Introdução, e procura descrever, esclarecer e justificar o trabalho, os seus objetivos, bem como a sua importância. É composto por quatro secções, nomeadamente: Enquadramento (1.1), Motivação do projeto (1.2), Finalidade do trabalho e Síntese do método de investigação (1.3), e Estrutura do documento (1.4).

O Capítulo 2 corresponde à Revisão da literatura, cujo objetivo é analisar o que existe na literatura sobre as temáticas em análise nesta dissertação. Este capítulo é composto por sete secções: Desenvolvimento de Sistemas de Informação (2.1), Ciclo de Vida Preditivo (2.2), Ciclo de Vida Ágil (2.3), Ciclo de Vida Iterativo e Incremental (2.4), *Agile* sob Diferentes Perspetivas (2.5), Implantação de Tecnologias da Informação (2.6), *Method Engineering* (2.7), Trabalhos Relacionados (2.8.), e Reflexão Crítica sobre o Estado da Arte (2.9).

No Capítulo 3 é descrito o Plano de Investigação utilizado, sendo composto por quatro secções: Problema (3.1), Descrição e Justificação do Método de Investigação (3.2), Processo de Investigação (3.3), e Estratégia de Pesquisa (3.4).

Segue-se o Capítulo 4, onde são apresentados os resultados obtidos. Este capítulo é composto por quatro secções: Metodologia de Implementação Primavera (MIP) (4.1), *Agile* na Implantação de TI (4.2), AgileMIP (4.3), Aplicação e Avaliação do AgileMIP (4.4), e AgileMIP *versus* MIP (4.5).

Por último, o Capítulo 5 corresponde à Conclusão, onde é efetuada uma síntese e reflexão sobre o trabalho realizado, os seus principais contributos e limitações, e, finalmente, sugestões de trabalhos futuros a realizar.

2 Revisão da literatura

Neste capítulo é apresentada a revisão da literatura elaborada no âmbito desta dissertação. Primeiro, é apresentada uma breve introdução ao Desenvolvimento de Sistemas de Informação (2.1). Segue-se uma apresentação dos ciclos de vida em *waterfall* (2.2), ágil (2.3), e iterativo e incremental (2.4). Depois, é apresentado o *agile* sob diferentes perspetivas (2.5), o processo de implantação de Tecnologias da Informação (2.6), o processo de criação de métodos (2.7), os trabalhos relacionados (2.8), e uma reflexão crítica sobre o estado da arte (2.9).

2.1. Desenvolvimento de Sistemas de Informação

O Desenvolvimento de Sistemas de Informação (DSI) é interpretado de forma diferente por diferentes autores, sendo alguns exemplos o desenvolvimento interno de *software*, a consultoria e o *design* de artefactos para mercados futuros (Korpela et al., 2002).

Beynon-Davies (2016, p.58) define DSI como a ciência e arte de desenhar e elaborar, com economia e elegância, sistemas de informação computadorizados que suportam a atividade das organizações. Afirma também que os Sistemas de Informação (SI) são tipicamente incorporados nos sistemas de atividades humanas, reconhecendo a importância do *design* dos processos de trabalho e não só do *design* das questões tecnológicas.

Xia e Lee (2004) apresentam também uma definição de DSI, que corrobora a perspetiva anteriormente apresentada, afirmando que o DSI envolve a análise, *design* e implementação de aplicações e sistemas que suportam as operações de negócio num contexto organizacional.

Por sua vez, Korpela et al. (2002) definem DSI como um processo através do qual uma atividade coletiva de trabalho é facilitada por um novo meio de Tecnologia da Informação (TI), através da análise, *design*, implementação, introdução e suporte, bem como da gestão de processos.

Corroborando as definições apresentadas, Pee et al. (2010) classificam o DSI como um processo intensivo de conhecimento que requer conhecimento multifuncional, destacando-se a área do negócio (que impulsionará o novo SI) e de Tecnologias da Informação (necessário para potenciar e otimizar o novo SI) como elementos críticos.

Assim sendo, pode concluir-se que o DSI e, consequentemente, os projetos de DSI, representam atividades com elevada complexidade, proveniente de duas dimensões: organizacional e tecnológica (Xia e Lee, 2004; Benbya e McKelvey, 2006; Carugati, 2008).

Posto isto, podem ser definidos diferentes elementos que complementam, suportam e organizam o DSI, tais como:

A. Métodos: conjunto de etapas, técnicas, ferramentas e/ou *deliverables* necessários para alcançar um objetivo (Odell, 1996). Carateriza-se pela sua orientação para objetivos, abordagem sistemática, existência de princípios (de *design*, construção ou estratégicos) e pela possibilidade de reutilização (Braun et al., 2005);
B. Metodologias: conjunto de métodos aplicados para desenvolver sistemas de informação (Odell, 1996);
C. Modelos: representação de um sistema com abstração de um problema em estudo (Wijers, 1991);
D. Processos: forma de fazer algo (Henderson-Sellers, 2006);
E. Ciclos de vida: conjunto de fases realizadas num projeto, desde que se inicia até ao seu término (PMI, 2017). Também conhecidos como abordagens (Agile Alliance e PMI, 2017).
F. Técnicas: constitui uma forma particular de realizar uma atividade ou processo (Beynon-Davies, 2016);
G. Ferramentas: meio para suportar o processo de desenvolvimento (Brinkkemper, 1996), tais como *software*, *hardware* ou outros recursos de comunicação (Beynon-Davies, 2016).

Existem diversos métodos, modelos e *frameworks* para a organização do DSI, orientados para as suas diferentes dimensões.

No PMBOK (PMI, 2017) os ciclos de vida são agrupados em quatro categorias, descritos nas secções seguintes, que são: ciclos de vida preditivos (secção 2.2), ciclos de vida ágeis (2.3), ciclos de vida iterativos e incrementais (secção 2.4) e, por fim, ciclos de vida híbridos.

Os ciclos de vida híbridos combinam caraterísticas de diferentes ciclos de vida, dos quais são exemplos o preditivo, iterativo, incremental e ágil (Agile Alliance e PMI, 2017).

Este tipo de ciclo é utilizado quando um projeto tem associada alguma incerteza e risco, mas também caraterísticas que beneficiam de um fluxo de trabalho mais controlado (PMI, 2017). Por exemplo, elementos que sejam claros, bem definidos ou com caraterísticas fixas ao longo do projeto, são geridos tendo em conta um ciclo de vida preditivo, enquanto os restantes são geridos numa perspetiva mais adaptável (PMI, 2017), conforme as necessidades do projeto.

2.2. Ciclo de vida preditivo

O ciclo de vida preditivo, ou em waterfall, está associado a uma abordagem mais clássica do DSI e possui um conjunto de caraterísticas próprias que o diferenciam dos restantes ciclos de vida (PMI, 2017; Agile Alliance e PMI, 2017):

- Identificação *a priori* de todos os requisitos;
- Realização de um planeamento exaustivo do projeto, *a priori*, com base nos requisitos conhecidos;
- Cada etapa do ciclo de vida é realizada apenas uma vez, de forma sequencial;
- Os requisitos do projeto são fixos;
- Apenas é realizada uma entrega no projeto, tipicamente, no seu término;
- A mudança é evitada ao máximo, não sendo bem aceite;
- Envolvimento dos *stakeholders* ocorre apenas em *milestones* específicos;
- O risco, âmbito do projeto, cronograma e os custos são controlados através de um planeamento detalhado, e estabelecidos nas etapas iniciais do ciclo de vida.

A. Modelo *waterfall*

O melhor exemplo do ciclo de vida preditivo é o modelo em *waterfall*, sendo considerado a base da maior parte dos modelos existentes (Fitzgerald, 2000). Foi desenvolvido em 1970 por Winston W. Royce e amplamente utilizado nos anos (19)70 e (19)80, até aos dias de hoje.

Este modelo preconiza um fluxo sequencial, o que significa que uma etapa segue outra e que uma nova etapa apenas se inicia depois de a anterior terminar. Significa, também, que existe uma ordem predefinida para a realização dessas etapas, a qual não pode ser alterada.

As etapas são, tipicamente, o estudo da viabilidade do projeto, análise, conceção, implementação, e, por fim, revisão e manutenção (Avison e Fitzgerald, 2006).

Stoica et al. (2013) discriminam as etapas do modelo *waterfall*, como sendo: análise e definição de requisitos, conceção, implementação, teste, disponibilização e manutenção.

Royce (1970), o proponente do modelo *waterfall*, identifica, conforme representado na Figura 1, que as etapas deste modelo são: levantamento de requisitos (perceber as funcionalidades, propósito e objetivo da aplicação informática), *design* preliminar (visão global da aplicação, descrição da sua infraestrutura e dos seus procedimentos operacionais), análise (decisão das caraterísticas técnicas do produto), *design* final da aplicação e elaboração de testes, desenvolvimento (produção da aplicação informática), teste (identificação dos problemas da aplicação), e, por fim, operações (elaboração de manuais de utilizador).

Adaptado de: (Royce, 1970, p.329)

Figura 1: Modelo *Waterfall*

A avaliação do produto desenvolvido ocorre apenas na etapa de testes, onde frequentemente se sente a necessidade de retornar às etapas de levantamento de requisitos ou de *design* (Matkovic e Tumbas, 2010). Tendo isto em conta, Royce (1970) acrescentou o fator *feedback*, que permite a interação da etapa atual com a etapa anterior, mas nunca com as mais afastadas.

Desta forma, mostra-se importante refletir sobre as forças e limitações deste modelo. Para isto, foi realizada uma análise na literatura, representada nas Tabelas 1 e 2, e descrita detalhadamente no Apêndice A.1.

Na Tabela 1 são apresentadas as principais forças do modelo *waterfall* encontradas na literatura. Através da análise da tabela é possível perceber que as suas principais forças são o facto de ser um modelo bem estruturado, documentado e extensivamente testado, e também o facto de ser fácil de perceber, utilizar e coordenar.

Outras forças estão relacionadas com a existência de um maior controlo do projeto, o uso de documentação *standard*, a identificação *a priori* de requisitos e a sua adequação a projetos de pequena dimensão.

Tabela 1: Forças do Modelo *Waterfall*

Referências / Forças	(McConnell, 1996)	(Roque,1998)	(Avison e Fitzgerald, 2006)	(Matkovic e Tumbas, 2010)	(Balaji e Murugaiyan, 2012)	(Wells, 2012) *	(Stoica et al., 2013)	(Sarker et al., 2015)	(Eason, 2016)	Total de referências
Modelo bem documentado e estruturado	X			X	X		X			4
Fácil de perceber, utilizar e coordenar					X		X	X	X	4
Modelo extensivamente testado e experimentado		X	X		X					3
Existência de um nível levado controlo		X	X							2
Utilização de documentação *standard*			X				X			2
Identificação dos requisitos *a priori*					X				X	2
Adequação a projetos de pequena dimensão							X	X		2
Adequação a projetos complexos e bem definidos	X									1

Referências / Forças	(McConnell, 1996)	(Roque,1998)	(Avison e Fitzgerald, 2006)	(Matkovic e Tumbas, 2010)	(Balaji e Murugaiyan, 2012)	(Wells, 2012) *	(Stoica et al., 2013)	(Sarker et al., 2015)	(Eason, 2016)	Total de referências
Permite avaliar o progresso no final de cada etapa		X								1
Garante a formação de todos os utilizadores			X							1
Inclui uma etapa de teste				X						1
Os indivíduos de uma equipa são facilmente substituídos				X						1
Cada etapa é finalizada num período específico de tempo					X					1
É um modelo linear					X					1
Extremamente eficaz, eficiente e confiável em termos de engenharia						X				1

O símbolo * significa que se trata de um artigo com evidência empírica

Relativamente às limitações do modelo *waterfall* (Tabela 2), duas principais limitações foram identificadas: inflexibilidade e a dificuldade em voltar atrás nas etapas. Outras limitações são a utilização excessiva de documentação, a visualização de resultados ou de um produto funcional (tipicamente) no final do projeto, a identificação de erros (que apenas ocorre na etapa de testes, no final do projeto), a implicação de custos elevados, e, também, o facto de existir alguma dificuldade na adaptação à incerteza.

Tabela 2: Limitações do Modelo *Waterfall*

Referências / Limitações	(McConnell, 1996)	(Roque, 1998)	(Avison eFitzgerald, 2006)	(Matkovic e Tumbas, 2010)	(Balaji e Murugaiyan, 2012)	(Wells, 2012) *	(Stoica et al., 2013)	(Sarker et al., 2015)	(Eason, 2016)	Total de referências
Inflexibilidade	X	X	X	X	X		X		X	7
Dificuldade em voltar atrás entre etapas	X	X			X		X	X		5
Documentação excessiva	X	X	X						X	4
Visualização de resultados/produtos funcional (tipicamente) apenas no final do projeto		X		X			X	X		4
Identificação de erros apenas ocorre no final do projeto e dificuldade na sua correção		X		X	X		X			4

Referências / Limitações	(McConnell, 1996)	(Roque, 1998)	(Avison eFitzgerald, 2006)	(Matkovic e Tumbas, 2010)	(Balaji e Murugaiyan, 2012)	(Wells, 2012) *	(Stoica et al., 2013)	(Sarker et al., 2015)	(Eason, 2016)	Total de referências
Custos elevados (tempo, dinheiro, ...)		X		X			X			3
Dificuldade na adaptação à incerteza				X		X		X		3
Dificuldade na identificação de requisitos *a priori*	X		X							2
Desenvolvimento de software não ambicioso e obsoleto		X	X							2
Riscos e níveis de incerteza elevados								X	X	2
Pouco envolvimento do cliente no processo de desenvolvimento	X									1
Insatisfação dos *stakeholders*			X							1
Impossibilidade de realizar iterações				X						1
Processo de desenvolvimento longo				X						1
Não adequado para projetos longos, complexos, orientados a objetos e onde existe uma probabilidade de mudança de requisitos								X		1

O símbolo * significa que se trata de um artigo com evidência empírica

Durante a realização desta análise, dois elementos foram identificados, simultaneamente, como forças e limitações: documentação e requisitos. No caso dos requisitos, acontece porque, para alguns autores (Balaji e Murugaiyan, 2012; Eason, 2016), a identificação de requisitos *a priori* permite uma identificação clara do trabalho a realizar, enquanto que para outros (McConnell, 1996; Avison e Fitzgerald, 2006), representa uma estrutura rígida e inflexível, que não permite a adaptação à mudança.

O mesmo acontece com a documentação: é vista como uma força no sentido de o trabalho a realizar para cada *feature* ser claro (Avison e Fitzgerald, 2006; Stoica et al., 2013), mas também como uma limitação, uma vez que requer um grande esforço e um elevado nível de detalhe (McConnel, 1996; Roque, 1998; Avison e Fitzgerald, 2006; Eason, 2016).

2.3. Ciclo de vida ágil

O ciclo de vida ágil, ou adaptativo ou orientado à mudança, consiste num conjunto de métodos, cujo objetivo é ajudar as equipas a pensar, trabalhar e tomar decisões eficientemente (Stellman e Greene, 2014). Tendo isto em conta, existem um conjunto de caraterísticas que diferenciam este ciclo de vida dos restantes (PMI, 2017; Agile Alliance e PMI, 2017):

- É iterativo e incremental;
- O seu objetivo é maximizar o valor para o cliente;

- Orientação para o cliente;
- Identificação de requisitos progressiva;
- Atualização frequente dos requisitos através de cada iteração;
- Entrega frequente, mais rápida, e realizada através de pequenos incrementos;
- Incorporação da mudança imediata;
- Rápida resposta à mudança;
- Envolvimento contínuo dos *stakeholders* do projeto;
- Obtenção frequente de *feedback* do cliente/ *stakeholders*/ utilizadores;
- O risco e o custo são controlados conforme o aparecimento de requisitos e restrições;
- Possibilidade de melhorar um trabalho;
- Levantamento dinâmico de requisitos;
- Promove e incentiva à colaboração e participação ativa de todos os elementos de um projeto.

A. *Scrum*

O *Scrum* é um bom exemplo de um ciclo de vida ágil, uma vez que é a *framework* mais utilizada, de acordo com o 13.º relatório anual sobre o estado do *agile* (CollabNet e VersionOne, 2019), onde 54% das organizações participantes utiliza *Scrum* ou uma abordagem híbrida que inclui *Scrum* (18%).

Foi proposto em 1995 por Ken Schwaber e Jeff Sutherland e permite a utilização de diversos processos e técnicas que auxiliam o desenvolvimento de produtos complexos (Schwaber, 2009).

Apesar de amplamente utilizado no desenvolvimento de *software*, o seu âmbito de aplicação é muito mais abrangente, podendo ser utilizado em qualquer projeto (Schwaber e Sutherland, 2017).

Os princípios do *Scrum* assentam sobre três pilares: transparência (todos os aspetos dos processos que afetam um resultado devem ser visíveis e conhecidos), inspeção (todos os aspetos do processo devem ser inspecionados com frequência por forma a serem detetadas anomalias e variações anormais) e adaptação (ajuste do processo quando existem desvios anormais) (Schwaber, 2009).

Paralelamente aos seus pilares, são também definidos um conjunto fundamental de elementos a possuir numa equipa, que são:

- *Product owner*: responsável pelo que será desenvolvido, em que ordem, e pelo sucesso do produto a entregar ou a manter (Rubin, 2012);
- Equipa de desenvolvimento: equipa auto-organizada e multifuncional (Schwaber e Sutherland, 2017), responsável por decidir como alcançar o objetivo estipulado pelo *product owner* (Rubin, 2012);

- *Scrum master*: responsável por garantir o entendimento do *Scrum*, as suas práticas, regras e valores (Schwaber e Sutherland, 2017), por auxiliar as equipas a melhorar o seu desempenho e organização, e por remover impedimentos ao seu trabalho (Rubin, 2012).

São também identificadas um conjunto de atividades ou cerimónias:

A. Reunião de planeamento da *sprint*: atividade onde a equipa se reúne para identificar o objetivo da *sprint*, determinar o que será entregue e quais os itens do *backlog* do produto que estão alinhados com o objetivo definido (Rubin, 2012). Esta atividade, tipicamente, tem alocadas oito horas para a sua realização (numa *sprint* de um mês) ou 5% da dimensão da *sprint*, sempre que esta seja de menor dimensão (Schwaber, 2009).
B. Revisão da *sprint*: os *stakeholders* e a equipa verificam e validam, no final da *sprint*, o incremento da *sprint* (Rubin, 2012). O objetivo é avaliar o incremento, e atualizar o *backlog* (Schwaber e Sutherland, 2017).
C. Retrospetiva da *sprint*: a equipa do projeto avalia os processos utilizados para criar o incremento (Rubin, 2012). O objetivo é a autoavaliação da equipa (identificar o que correu bem, o que correu mal, impedimentos, etc.), e a definição de um plano de melhoria a aplicar na próxima *sprint* (Schwaber e Sutherland, 2017).
D. Reunião de planeamento da *release*: consiste em ajustar o âmbito, datas e o orçamento às entregas incrementais (Rubin, 2012). O objetivo é perceber como se pode criar um incremento do produto com valor e que satisfaça o cliente (Schwaber, 2009).
E. Reuniões diárias: evento diário com duração de 15 minutos, realizado com a equipa de desenvolvimento, onde é atualizado o progresso do trabalho (Schwaber, 2009).

Complementarmente, os artefactos considerados essenciais no *Scrum* são os seguintes:

A. *Backlog* do produto: lista de itens priorizados ou ordenados do trabalho a realizar num projeto, que está continuamente em desenvolvimento, atualização, revisão, e refinamento, conforme as condições do negócio mudam, e o conhecimento sobre o projeto e o produto mudam (Rubin, 2012). O *product owner* é responsável por manter e comunicar este documento (Schwaber e Sutherland, 2017).
B. *Backlog* da *sprint*: descreve, através de um conjunto detalhado de tarefas, como os elementos selecionados do *backlog* serão desenvolvidos durante uma *sprint* (Rubin, 2012) (Schwaber, 2009).
C. Incremento do produto: resultado de uma *sprint* (Rubin, 2012).

Posto isto, é possível identificar as forças e limitações do *Scrum*, conforme apresentado nas Tabelas 3 e 4 e, em maior detalhe, no Apêndice A.6.

Através da análise da Tabela 3, pode concluir-se que as principais forças do *Scrum* são a obtenção rápida de resultados, o aumento da colaboração e comunicação, e a melhoria da aprendizagem e motivação da equipa. Outras forças que também sobressaem na análise são relativas à entrega ser mais rápida, frequente, contínua e iterativa, a obtenção contínua de *feedback* dos clientes, a maior satisfação dos *stakeholders,* e à existência de maior transparência sobre o projeto.

Tabela 3: Forças do *Scrum*

Referências / Forças	(Amlani, 2012)	(Overhage e Schlauderer, 2012) *	(Rubin, 2012)	(Mahalakshmi e Sundararajan, 2013)	(Law e Lárusdóttir, 2015) *	(Streule et al., 2016) *	(Obrutsky, 2016)	(Azanha et al., 2017) *	Total de referências
Obtenção de melhores resultados mais rapidamente	X	X	X				X		4
Aumento da colaboração e comunicação		X			X	X		X	4
Melhoria da aprendizagem e motivação da equipa		X			X	X		X	4
Entrega rápida, iterativa, frequente e contínua	X				X			X	3
Feedback contínuo do cliente	X	X					X		3
Maior satisfação dos *stakeholders* (cliente, equipa do projeto, etc.)			X	X				X	3
Maior transparência sobre o projeto		X			X	X			3
Otimização de recursos (tempo, dinheiro, etc.)	X							X	2
Permite alcançar o sucesso do projeto onde a documentação e a identificação de requisitos são difíceis de elaborar e onde existe uma grande quantidade de complexidade envolvida	X		X						2
Aceitação e facilidade na realização mudanças	X			X					2
Flexibilidade e adaptação à mudança		X					X		2
Maior controlo sobre o projeto				X				X	2
Realização de *sprints* de curta duração				X	X				2
Modelo leve, sem grande controlo	X								1
Fácil de medir a produtividade individual	X								1
Identificação e correção rápida de problemas	X								1
Funciona com qualquer tecnologia	X								1
Permite o levantamento de requisitos depois da entrega de um *deliverable*							X		1
Melhoria do retorno de investimento			X						1
Redução de custos			X						1
Mais felicidade			X						1

Referências / Forças	(Amlani, 2012)	(Overhage e Schlauderer, 2012) *	(Rubin, 2012)	(Mahalakshmi e Sundararajan, 2013)	(Law e Lárusdóttir, 2015) *	(Streule et al., 2016) *	(Obrutsky, 2016)	(Azanha et al., 2017) *	Total de referências
Aumento da qualidade do produto				X					1
Providencia de melhores estimativas				X					1
Ideal para mudanças rápidas				X					1
Priorização de tarefas				X					1
O trabalho a realizar numa *sprint* não é modificado				X					1
Permite ajustar o âmbito do projeto				X					1
A estimativa do trabalho é mais fácil de realizar				X					1
Menor tempo de colocação do produto no mercado		X							1
Melhor perceção dos requisitos		X							1
Melhor fluxo informacional						X			1
Melhoria contínua					X				1

O símbolo * significa que se trata de um artigo com evidência empírica

Relativamente às limitações do *Scrum*, identificadas na Tabela 4, não existe um consenso, todavia, é possível identificar como principal limitação, o impacto da desistência de um elemento de uma equipa ou a fraca cooperação entre os seus diversos elementos.

Apesar de não se identificar um consenso sobre as limitações do *Scrum*, é possível identificar um elemento comum na maior parte das limitações encontradas na literatura: **pessoas**. Várias são as referências relacionadas com as pessoas, o seu comportamento, e a sua postura, destacando-se a dependência do empenho, coordenação, colaboração, trabalho em equipa, e confiança entre os elementos de uma equipa e entre equipas.

Tabela 4: Limitações do *Scrum*

Referências / Limitações	(Amlani, 2012)	(Overhage e Schlauderer, 2012) *	(Mahalakshmi e Sundararajan, 2013)	(Streule et al., 2016) *	(Obrutsky, 2016)	Total de referências
A desistência de um membro da equipa ou a fraca cooperação entre os seus elementos, pode levar ao fracasso do projeto	X		X		X	3

Referências / Limitações	(Amlani, 2012)	(Overhage e Schlauderer, 2012) *	(Mahalakshmi e Sundararajan, 2013)	(Streule et al., 2016) *	(Obrutsky, 2016)	Total de referências
Não funciona bem para equipas grandes	X				X	2
A gestão/controlo da qualidade é difícil de implementar se a equipa de testes não estiver disponível em cada *sprint*	X				X	2
Não existência de uma data explícita para o termo do projeto	X					1
Dificuldade na estimativa da duração e custo do projeto	X					1
Dependência do emprenho dos diferentes elementos da equipa Dependence on the commitment of the different team members	X					1
Não funciona bem para projeto de grande dimensão	X					1
Necessidade de profissionais com experiência ou com grandes competências	X					1
Necessidade de confiança entre os membros da equipa	X					1
Existência de pouca documentação			X			1
O trabalho em equipa é essencial			X			1
Aumento da complexidade		X				1
Necessidade de maior disciplina		X				1
Dificuldade na iniciação do uso da *framework*				X		1
Identificação pouco clara da liderança do projeto				X		1

O símbolo * significa que se trata de um artigo com evidência empírica

2.4. Ciclos de vida iterativos e incrementais

Os ciclos de vidas iterativos e incrementais situam-se entre os ciclos de vida preditivos e os ciclos de vida ágeis. O ciclo iterativo encontra-se mais próximo do ciclo de vida preditivo, enquanto o incremental se aproxima mais dos ciclos de vida ágeis (PMI, 2017).

Os ciclos de vida iterativos caraterizam-se por (PMI, 2017; Agile Alliance e PMI, 2017):

- Obtenção de *feedback* sobre trabalho não acabado, e a sua modificação e melhoria;
- Consideração de apenas uma entrega, no final do projeto;
- Levantamento dinâmico de requisitos;
- Realização de múltiplas iterações, até ao produto possuir a máxima qualidade.

Por sua vez, os ciclos incrementais caraterizam-se por (PMI, 2017; Agile Alliance e PMI, 2017):

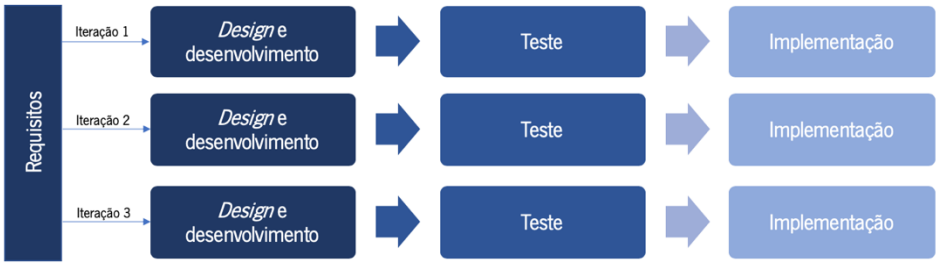
- Disponibilização de *deliverables* funcionais em cada iteração, os quais o cliente pode utilizar imediatamente em ambiente de produção;
- Repetição das etapas do ciclo de vida para cada incremento;
- Levantamento dinâmico de requisitos;

- Cada iteração representa um incremento finalizado.

A. Modelo incremental

O modelo incremental foi proposto por volta de 1990, e o seu objetivo é a melhoria do produto através da entrega de vários incrementos do produto, a redução do tempo de desenvolvimento, e a melhoria do processo de mudança (Avison e Fitzgerald, 2006).

Este modelo, conforme representado na Figura 2, envolve a realização de múltiplas iterações, compostas pelas mesmas etapas (Stoica et al., 2013): levantamento de requisitos, desenvolvimento, teste e implementação, resultando sempre um novo incremento do produto.



Adaptado de: (Stoica et al., 2013

Figura 2: Modelo Incremental

As suas principais forças, representadas na Tabela 5 e, em maior detalhe, no Apêndice A.2, estão relacionadas com os seus ciclos. Assim, as suas principais forças são a entrega de um incremento do produto totalmente funcional em cada iteração e a obtenção de *feedback* do cliente ao longo de todo o processo de desenvolvimento, o que revela foco no cliente.

Outras forças identificadas são a priorização das tarefas a realizar, a realização do trabalho através de pequenos incrementos, a redução de custos nas entregas iniciais, e a maior facilidade na gestão de riscos.

Tabela 5: Forças do Modelo Incremental

Forças \ Referências	(Bhuvaneswari e Prabaharan, 2013)	(Stoica et al., 2013)	(Alshamrani e Bahattab, 2015)	(Sarker et al.,2015)	Total de referências
Cada entrega representa um produto funcional		X	X	X	3
Obtenção do *feedback* do cliente ao longo de todo o processo de desenvolvimento	X	X		X	3
Priorização de tarefas			X	X	2
Realização do trabalho através de pequenos incrementos (o que propicia a criação de valor desde cedo no projeto)	X		X		2
Redução de custo nas entregas iniciais		X		X	2
Maior facilidade na gestão de riscos		X		X	2

Referências / Forças	(Bhuvaneswari e Prabaharan, 2013)	(Stoica et al., 2013)	(Alshamrani e Bahattab, 2015)	(Sarker et al.,2015)	Total de referências
Baixo custo de mudança de requisitos e/ou âmbito		X		X	2
Melhoria e aprendizagem contínua			X		1
Maior rapidez na entrega do produto			X		1
Redução do risco de falha e/ou mudança de requisitos			X		1
Existência de protótipos funcionais		X			1
Maior facilidade em testar e fazer *debug*		X			1
Melhor uso dos recursos através de uma clara definição do incremento	X				1
Deteção de problemas desde cedo	X				1

O símbolo * significa que se trata de um artigo com evidência empírica

Relativamente às limitações do modelo incremental, identificadas na Tabela 6 e no Apêndice A.2, destacam-se duas limitações: a necessidade de elaboração de um planeamento e *design* detalhados, e a necessidade de definição, *a priori*, de todas as funcionalidades da aplicação informática a desenvolver.

Existem também dois autores (Stoica et al., 2013; Sarker et al., 2015) que identificam como limitação a existência de custos mais elevados, quando comparando este modelo com o modelo em *waterfall*.

Tabela 6: Limitações do Modelo Incremental

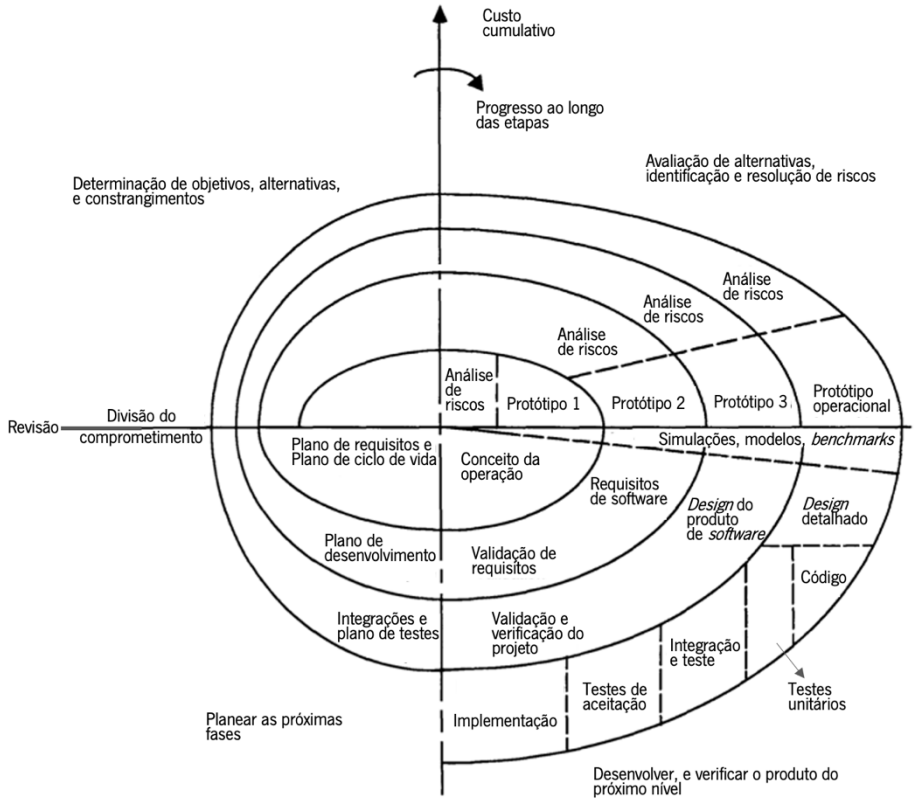
Referências / Limitações	(Bhuvaneswari e Prabaharan, 2013)	(Stoica et al., 2013)	(Alshamrani e Bahattab, 2015)	(Sarker et al., 2015)	Total de referências
Requer um bom planeamento de *design*		X	X	X	3
Requer uma definição de todas as funcionalidades do produto *a priori*		X	X	X	3
Custos mais elevados		X		X	2
Não permite a realização de iterações no mesmo incremento			X		1
Pode tornar-se num ciclo de "*code and repair*"		X			1
Requer muita documentação	X				1
Segue um conjunto predefinido de processos	X				1
Define os incrementos baseados na função e nas dependências	X				1
Requer um maior envolvimento do cliente	X				1
A integração entre iterações pode ser um problema	X				1

O símbolo * significa que se trata de um artigo com evidência empírica

B. Modelo em espiral

O modelo em espiral foi proposto em 1988 por Barry W. Boehm, sendo o modelo que mais contribuiu para o desenvolvimento dos ciclos de vida iterativos (Matkovic e Tumbas, 2010).

Possui duas dimensões (Boehm, 1988): radial (representa os custos ao longo das diferentes etapas) e angular (representa o progresso ao longo do tempo).



Adaptado de: (Boehm, 1988)

Figura 3: Modelo em Espiral

Os seus ciclos são, conforme representado na Figura 3, sempre iniciados com o planeamento, que consiste na identificação dos objetivos e restrições do projeto (tempo, custos, etc.), e na definição da abordagem de implementação para o incremento a desenvolver (Boehm, 1988).

Após o planeamento, segue-se a realização da análise de riscos, representada no segundo quadrante da Figura 3. Nesta análise, são identificados os elementos da iteração com maior risco e nível de incerteza, e são (re)avaliados os objetivos estabelecidos. São também aplicadas técnicas para minimizar os riscos identificados na análise, tais como prototipagem, simulação, *benchmarking* e/ou questionários (Boehm, 1988). Depois de aprovada a versão final do protótipo, inicia-se a etapa de desenvolvimento, que é realizada seguindo uma abordagem em *waterfall*.

Após realização da etapa de desenvolvimento, ocorre o planeamento da iteração seguinte, cujo foco pode incidir sobre o produto como um todo ou sobre uma parte específica do mesmo (um incremento).

Para terminar a iteração, ocorre a revisão do trabalho realizado, sendo envolvidos os principais atores ou organizações na conceção do produto ou incremento do produto. Aqui, ocorre uma análise do trabalho realizado, incluindo do planeamento sobre a próxima iteração.

Em suma, o modelo em espiral tem associada a premissa de que um conjunto de objetivos podem ser melhorados através da realização de múltiplos ciclos, onde o trabalho realizado é sempre testado (no terceiro quadrante do modelo, conforme representado na Figura 3) (Boehm, 1988).

Relativamente às forças do modelo em espiral, representadas na Tabela 7 (e em maior detalhe no Apêndice A.3), pode concluir-se que assentam no desenvolvimento de *software* funcional desde cedo no projeto e num período curto de tempo, e na valorização da análise de riscos. Outras forças, são a sua adequação a projetos grandes, críticos e de elevado risco, e, também a flexibilidade presente na etapa de engenharia.

Tabela 7: Forças do Modelo em Espiral

Forças \ Referências	(Matkovic e Tumbas, 2010)	(Bhuvaneswari e Prabaharan, 2013)	(Alshamrani e Bahattab, 2015)	(Uniyal e Kalia, 2016) *	Total de referências
Desenvolvimento de *software* funcional desde cedo no projeto e num período curto de tempo	X	X	X		3
Valorização da análise de riscos		X	X	X	3
Flexibilidade na etapa de engenharia (por exemplo, requisitos) Flexibility in the stage of engineering (for example, requirements)	X		X		2
Bom para projetos grandes, críticos e com elevado risco		X		X	2
Identificação dos riscos desde o início do projeto			X	X	2
Possibilidade de combinar diferentes abordagens de desenvolvimento de *software*	X				1
Possibilidade de analisar o risco a qualquer momento em qualquer nível de abstração	X				1
Traduz uma abordagem sistemática do modelo em *waterfall*, com possibilidade de iteração	X				1
Forte controlo da documentação			X		1
Feedback constante, desde cedo, pelos utilizadores			X		1

O símbolo * significa que se trata de um artigo com evidência empírica

As limitações do modelo em espiral são apresentadas na Tabela 8, e, em maior detalhe, no Apêndice A.3. Através da análise da tabela, é possível concluir que a maior limitação deste modelo está relacionada com uma das suas principais forças: a análise de riscos.

Conforme identificado anteriormente, a análise de riscos é bastante valorizada neste modelo, mas, a sua concretização, requer conhecimento detalhado sobre as suas especificações, sendo esta, de acordo com a análise realizada, uma das suas principais limitações.

Outra limitação, também relacionada com a análise de riscos, é o seu impacto no sucesso do projeto, uma vez que uma falha nesta análise, possui um impacto significativo no sucesso do projeto.

Tabela 8: Limitações do Modelo em Espiral

Referências / Limitações	(Matkovic e Tumbas, 2010)	(Bhuvaneswari e Prabaharan, 2013)	(Alshamrani e Bahattab, 2015)	(Uniyal e Kalia, 2016) *	Total de referências
Requer conhecimento específico para a realização da análise de riscos	X	X	X		3
Não adequado a produtos ou projetos pequenos	X	X	X		3
Uma falha na análise de riscos tem um impacto significativo no sucesso do projeto	X	X			2
Tem elevados custos associados		X	X		2
Falta de *standards* para este modelo	X				1
Necessidade de maior uniformidade e consistência no desenvolvimento	X				1
A quantidade de documentação necessárias nas etapas intermediárias torna o modelo mais complexo			X		1

O símbolo * significa que se trata de um artigo com evidência empírica

C. *Unified Process Model* (UP)

O *Unified Process* (UP) foi proposto por Ivar Jacobson, Grady Booch e James Rumbaugh em 1999 (Matkovic e Tumbas, 2010). É um modelo de processo iterativo e incremental, o que significa que o projeto é dividido em pequenos projetos (iterações) que entregam funcionalidades do sistema em incrementos (Arlow e Neustadt, 2002).

Pode ser dividido em duas dimensões: tempo e conteúdo (Matkovic e Tumbas, 2010). A primeira dimensão, relativa ao tempo, representa o ciclo de vida do projeto (Arlow e Neustadt, 2002), e é composta por quatro etapas (Osis e Donis, 2017):

- Iniciação: etapa de planeamento onde se define a base do projeto, o seu âmbito, riscos, custos, objetivos e visão;
- Elaboração: realiza-se o levantamento de requisitos da aplicação informática, uma análise de riscos, a definição da arquitetura informacional da aplicação informática, desenho da aplicação, e é definido o método de implantação;

- Construção: finalização e melhoria do *design* do produto e desenvolvimento do produto;
- Transição: entrega do incremento do produto ao cliente, e implantação no seu contexto organizacional. Poderá incluir atividades como a migração de dados ou formação de utilizadores.

A segunda dimensão, relativa ao conteúdo, é composta por seis atividades de engenharia (Rational Software, 2011) (Osis e Donis, 2017):

- Modelação do negócio: modelação dos objetos e processos de negócio;
- Requisitos: identificação dos requisitos funcionais e não funcionais da aplicação informática;
- Análise e *design*: *design* da aplicação e da sua arquitetura informacional;
- Implementação: desenvolvimento da aplicação informática;
- Teste: validação das funcionalidades desenvolvidas;
- *Deployment*: implantação da aplicação informática.

Na segunda dimensão, existem ainda três atividades auxiliares, que são (Rational Software, 2011):

- Configuração e gestão da mudança: gestão e controlo da mudança dos artefactos no projeto;
- Gestão do projeto: procura equilibrar os objetivos do projeto, gerir o risco e ultrapassar impedimentos à entrega do incremento que vai de encontro às necessidades dos clientes e dos utilizadores;
- Ambiente: providenciar o ambiente de desenvolvimento necessário (ferramentas e processos) para suportar a equipa de desenvolvimento.

Relativamente à análise das forças e limitações do UP, representadas nas Tabelas 9 e 10, e, em maior detalhe, no Apêndice A.4., pode dizer-se que a informação existente é focada nas extensões ou adaptações do UP, maioritariamente no *Rational Unified Process* (RUP). O RUP é, no entanto, bastante similar ao UP (Arlow e Neustadt, 2002), pelo que as suas forças e limitações foram também consideradas nesta secção.

Na Tabela 9 são apresentadas as forças do UP. Através da sua análise, é possível concluir que as principais forças do UP assentam na melhoria da qualidade do produto, na melhoria da comunicação entre a equipa e com o cliente, e, também, na realização de entregas de incrementos funcionais do produto mais frequentemente e desde cedo no projeto.

Tabela 9: Forças do *Unified Process*

Forças \ Referências	(Rational Software, 1998)	(Kruchten, 2001)	(Zhou, 2009)	(Osorio et al., 2011) *	(Anwar, 2014)	Total de referências
Melhoria da qualidade do produto	X	X			X	3
Melhoria da comunicação da equipa e com o cliente		X		X	X	3
Entrega de um produto funcional com maior regularidade e mais cedo no projeto			X	X	X	3
Maior controlo do projeto		X		X		2
Os riscos são identificados e mitigados desde o início do projeto	X				X	2
A mudança é gerida explicitamente	X				X	2
Maior satisfação do cliente		X				1
Redução de custos e atrasos		X				1
Providencia todo o ciclo de vida de desenvolvimento de *software* (clássico)			X			1
Flexível e adaptável			X			1
Adequado a projeto de média de grande dimensão			X			1
Requer planeamento no início de cada iteração			X			1
Melhora a previsão de resultados na etapa de execução				X		1
Melhor ajuste aos requisitos do cliente				X		1
Maior nível de reutilização	X					1
Aprendizagem e melhoria contínua	X					1
Definição da estrutura do projeto *a priori*					X	1
Foco nas atividades mais importantes do projeto					X	1

O símbolo * significa que se trata de um artigo com evidência empírica

A maior limitação do UP encontrada na literatura, representada na Tabela 10, está relacionada com a quantidade de integrações realizadas durante o projeto, particularmente no final do projeto, onde é realizada uma integração global de todos os incrementos, tornando o processo bastante complexo e propício a erros.

Tabela 10: Limitações do *Unified Process*

Limitações \ Referências	(Zhou, 2009)	(Osorio et al., 2011) *	(Anwar, 2014)	Total de referências
Existência de pequenas integrações ao longo do projeto, com uma integração final aquando o término do projeto no final	X		X	2

Limitações \ Referências	(Zhou, 2009)	(Osorio et al., 2011) *	(Anwar, 2014)	Total de referências
Orientado a casos de uso	X			1
Elevado número de artefactos	X			1
Fraco suporte na gestão do projeto		X		1
Fraca integração dos *stakeholders* e do seu *feedback*		X		1
Requer customização tendo em conta a organização e a equipa, o que é um processo complexo e representa um grande custo e necessidade de profissionais com experiência nessa atividade		X		1

O símbolo * significa que se trata de um artigo com evidência empírica

2.5. *Agile* sob diferentes perspetivas

O "*agile*" foi inicialmente explorado no desenvolvimento de *software*, área à qual está tipicamente associado. Os seus primeiros passos ocorreram em 2001, aquando o lançamento do *Agile Manifesto* (Beck et al., 2001), onde são definidos os valores e princípios do *agile* para o desenvolvimento de *software*.

Esse conjunto de valores e princípios assentam na maior capacidade de resposta, maior visibilidade sobre o projeto, uma implementação mais rápida, maior flexibilidade, maior envolvimento do cliente, rápida entrega de funcionalidade, colaboração, e trabalho em equipa (Deloitte, 2020), os quais se podem transpor num *mindset*.

Na atualidade, o *agile* não é algo apenas relacionado com as Tecnologias da Informação (TI), incluindo situações de grande escala, de natureza complexa e que acontecem num elevado número de contextos diferenciados, que vão muito mais além do desenvolvimento de *software* (Axelos, 2018).

Pode, por isso, ser aplicado a diversas áreas do conhecimento, para além do desenvolvimento de *software*, tais como a consultoria, saúde, educação e manufatura (Agile Alliance e PMI, 2017).

Introduziu um elemento crucial, a agilidade, que consiste na capacidade de mudar e adaptar-se rapidamente às situações do ambiente (Bishop et al., 2018).

Por este motivo, cada vez mais as organizações procuram conhecer o conceito associado ao *agile,* (Deloitte, 2020), tentando integrá-lo nos seus métodos de trabalho e na sua forma de pensar, independentemente do seu ramo de atuação (público ou privado) ou da sua dimensão (Axelos, 2018).

No entanto, a transição de um paradigma clássico (tipicamente em *waterfall*) para um paradigma ágil, requer uma mudança na forma de pensar, que não é tão simples quanto descrito na literatura (Stellman e Greene, 2014). Todavia, é possível identificar três elementos cruciais para se alcançar o

sucesso na utilização de um paradigma ágil (Mohammed et al., 2013): pessoas (principal fator de sucesso), colaboração com o cliente, e adequada resposta à mudança.

A. *Agile* no desenvolvimento de *software*

Conforme já referido, o *agile* foi, desde cedo, amplamente explorado pela área de desenvolvimento de *software*. Os primeiros passos na história do *agile* foram dados em 2001, com o lançamento do *Agile Manifesto* (Beck et al., 2001), onde se encontram definidos os valores e princípios do *agile* para esta área.

Os valores definidos no *Agile Manifesto* são os seguintes (Beck et al., 2001):

1. "Indivíduos e interações mais do que processos e ferramentas": demonstra a valorização das pessoas e da comunicação cara-a-cara.
2. "*Software* funcional mais do que documentação abrangente": determina que é mais importante ter um incremento que funciona e que está de encontro com as necessidades do cliente, do que ter a aplicação informática extensivamente documentada e não funcional.
3. "Colaboração com o cliente mais do que negociação contratual": apela ao trabalho em equipa e à colaboração entre os diversos elementos do projeto.
4. "Responder à mudança mais do que seguir um plano": demonstra a importância de gerir e responder adequadamente à mudança, de se compreender que os planos se desatualizam rapidamente, e que é mais importante entregar um produto do que o seu planeamento.

Em acréscimo aos valores anteriormente explicitados, são também identificados doze princípios, que estão dependentes da ideia de "fazemos projetos para entregar valor" (Stellman e Greene, 2014), que são (Beck et al., 2001):

1. A maior prioridade é, desde as primeiras etapas do projeto, satisfazer o cliente através da entrega rápida e contínua de *software* com valor.
2. Aceitar alterações de requisitos, mesmo numa fase tardia do ciclo de desenvolvimento. Os processos ágeis potenciam a mudança em benefício da vantagem competitiva do cliente.
3. Fornecer frequentemente *software* funcional. Os períodos de entrega devem ser de poucas semanas a poucos meses, dando preferência a períodos mais curtos.
4. O cliente e a equipa de desenvolvimento devem trabalhar juntos, diariamente, durante o decorrer do projeto.
5. Desenvolver projetos com base em indivíduos motivados, dando-lhes o ambiente e o apoio de que necessitam, confiando que irão cumprir os objetivos.

6. O método mais eficiente e eficaz de passar informação para e dentro de uma equipa de desenvolvimento é através de conversa pessoal e direta.
7. A principal medida de progresso é a entrega de *software* funcional.
8. Os processos ágeis promovem o desenvolvimento sustentável. Os promotores, a equipa e os utilizadores deverão ser capazes de manter, indefinidamente, um ritmo constante.
9. A atenção permanente à excelência técnica e um bom desenho da solução aumentam a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho que não é feito – é essencial.
11. As melhores arquiteturas, requisitos e desenhos surgem de equipas auto-organizadas.
12. A equipa reflete regularmente sobre o modo de se tornar mais eficaz, fazendo os ajustes e adaptações necessárias.

Os valores e princípios apresentados, consistem no conteúdo do *Agile Manifesto*. Podem ser identificados princípios e valores que se relacionam diretamente com o desenvolvimento de *software*, tal como o valor 2, e os princípios 1, 3 e 7. No entanto, podem facilmente ser adaptados para qualquer outra área do conhecimento.

No *agile* são considerados elementos que nenhum outro ciclo de vida considera. Contudo, não é suficiente conhecer estes princípios e valores, é também necessário avaliar interações, relacionamentos, e comunicações dentro de uma equipa e organização (Stellman e Greene, 2014).

Tendo isto em conta, pode dizer-se que no *agile* as equipas, os indivíduos, e as suas caraterísticas são um fator crucial para o sucesso do projeto (Matkovic e Tumbas, 2010), bem como a colaboração com o cliente e a capacidade de resposta à mudança (Mohammed et al., 2013).

As suas forças e limitações foram estudadas pela CollabNet e VersionOne (2019), e foram descritas num relatório, elaborado anualmente com o objetivo de descrever o estado do *agile*. De acordo com o relatório, os principais benefícios aquando a adoção do *agile* são a capacidade de gerir a mudança de prioridades (69%), o aumento da visibilidade do projeto (65%), o maior alinhamento entre o negócio e as Tecnologias da Informação (64%), aumento da moral da equipa (64%), maior rapidez na entrega ao mercado (63%), e maior produtividade (61%).

Paralelamente a este estudo, foi realizado o levantamento das forças e limitações do *agile*, representadas na Tabela 11 e 12, respetivamente. Uma apresentação mais detalhada é apresentada no Apêndice A.5.

Assim sendo, as principais forças são a entrega de um produto de maior qualidade ao cliente (referido por 72% dos artigos), a capacidade de responder à mudança de requisitos (63% dos artigos) e a existência de comunicação direta entre o cliente e a equipa de desenvolvimento (45% dos artigos).

Tabela 11: Forças dos modelos ágeis

Referências / Forças	(Begel e Nagappan, 2007) *	(Laanti et al., 2011) *	(Balaji e Murugaiyan, 2012)	(McCormick, 2012)	(Kumar e Bhatia, 2012)	(Moniruzzaman e Hossain, 2013)	(Papatheocharous e Andreou, 2013) *	(Mohammed et al., 2013)	(Solinski e Peterson, 2016) *	(Uniyal e Kalia, 2016) *	(Kamei et al., 2017) *	Total de referências
Entrega de um produto de maior qualidade ao cliente	X	X		X	X	X	X			X	X	8
Habilidade de responder à mudança de requisitos		X	X	X	X	X	X			X		7
Comunicação direta entre o cliente e a equipa de desenvolvimento	X		X	X				X			X	5
Os testes ocorrem em cada iteração (melhor deteção e resolução de problemas)	X				X			X			X	4
Equipas mais satisfeitas	X						X	X	X			4
Maior visibilidade e transparência do projeto		X					X		X		X	4
No final de cada iteração, há uma versão funcional do produto (Tempo de mercado reduzido)					X	X	X	X				4
Imediata integração de mudanças	X	X		X								3
Entregas mais rápidas	X	X				X						3
Maior flexibilidade	X				X			X				3
Maior envolvimento e *feedback* do cliente	X					X		X				3
Entrega incremental, iterativa e evolutiva		X			X	X						3
Redução de custos, tempo de entrega e de desenvolvimento						X	X			X		3
Facilidade de interação, colaboração, partilha e comunicação		X								X	X	3
Maior produtividade, foco, eficiência e capacidade de previsão	X	X			X							3
Documentação na medida certa				X		X						2
Melhoria do desempenho					X		X					2
Priorização dos requisitos	X					X						2

Referências / Forças	(Begel e Nagappan, 2007) *	(Laanti et al., 2011) *	(Balaji e Murugaiyan, 2012)	(McCormick, 2012)	(Kumar e Bhatia, 2012)	(Moniruzzaman e Hossain, 2013)	(Papatheocharous e Andreou, 2013) *	(Mohammed et al., 2013)	(Solinski e Peterson, 2016) *	(Uniyal e Kalia, 2016) *	(Kamei et al., 2017) *	Total de referências
Adequado para projetos de pequena dimensão										X		1
Satisfação do cliente				X								1
Desenvolvimento do projeto através de pequenas iterações								X				1
Baseado nas pessoas e na sua criatividade								X				1
Equipas auto-organizadas						X						1
Conceção simples						X						1
Aumenta o valor do negócio, a sua visibilidade e a capacidade de adaptação						X						1
Aumento da probabilidade de sucesso						X						1
Facilidade de monitorizar e controlar o projeto											X	1

O símbolo * significa que se trata de um artigo com evidência empírica

Por fim, no que respeita às principais limitações do *agile* (Tabela 12), podem apontar-se a dificuldade na sua aplicação em projetos de grande escala (100% dos autores), a necessidade de profissionais seniores, com experiência ou especializados (60% dos autores), a falta de planeamento *a priori* (30% dos autores), e a errada perceção existente sobre o conceito *agile* (30% dos autores).

Tabela 12: Limitações dos modelos ágeis

Referências / Limitações	(Begel e Nagappan, 2007) *	(Balaji e Murugaiyan ,2012)	(McCormick, 2012)	(Kumar e Bhatia, 2012)	(Mohammed et al., 2013)	(Agrawal e t al., 2016) *	(Solinski e Peterson, 2016) *	(Uniyal e Kalia, 2016) *	(Kamei et al., 2017) *	Total de referências
Difícil de aplicar em projeto de grande escala	X	X	X	X	X		X	X		8
Requer profissionais seniores, com experiência ou especializados		X	X		X		X	X	X	6

Referências / Limitações	(Begel e Nagappan, 2007) *	(Balaji e Murugaiyan ,2012)	(McCormick, 2012)	(Kumar e Bhatia, 2012)	(Mohammed et al., 2013)	(Agrawal e t al., 2016) *	(Solinski e Peterson, 2016) *	(Uniyal e Kalia, 2016) *	(Kamei et al., 2017) *	Total de referências
Requires senior, experienced or specialized professionals										
Falta de planeamento a longo-prazo (previsibilidade)	X				X	X				3
Perceção errada sobre o conceito do *agile* (suposição de que não há regras ou documento; não há uma mudança na forma de pensar)	X	X			X					3
Requer demasiadas reuniões	X					X				2
Falta de precisão/planeamento antes de o projeto começar	X					X				2
Longos períodos de tempo podem ser atribuídos para o desenvolvimento de uma funcionalidade pequena				X						1
Interação com o cliente (o cliente pode não ter tempo suficiente para estar com a equipa)					X					1
As equipas ágeis devem estar localizadas no mesmo espaço físico por longos períodos de tempo					X					1
Dificuldade em identificar contributos individuais					X					1
Dificuldade em identificar como satisfazer os funcionários					X					1
Foco no processo de levantamento de requisitos e no desenvolvimento de código, e não no desenho do produto				X						1
Altos prazos de execução de testes				X						1
Baixa cobertura dos testes				X						1
Requer grande coordenação e comunicação com o gestor de projeto				X						1
Pode representar grandes custos				X						1
Perda da perspetiva global do projeto	X									1
Dificuldade em utilizar *user stories*									X	1

Referências / Limitações	(Begel e Nagappan, 2007) *	(Balaji e Murugaiyan ,2012)	(McCormick, 2012)	(Kumar e Bhatia, 2012)	(Mohammed et al., 2013)	(Agrawal e t al., 2016) *	(Solinski e Peterson, 2016) *	(Uniyal e Kalia, 2016) *	(Kamei et al., 2017) *	Total de referências
Dificuldade em trabalhar em equipas grandes									X	1
Interferência de *product owner* com competências técnicas									X	1
Dificuldade na aplicação e adaptação									X	1
Restrições financeiras/ de orçamento						X				1
Falta de documentação						X				1
Não segue as etapas dos modelos clássicos de desenvolvimento de *software*						X				1
Requer formação						X				1
Não é adequado para organizações de pequenas dimensões						X				1

O símbolo * significa que se trata de um artigo com evidência empírica

Um outro aspeto relevante relativo ao *agile,* e identificado no relatório anual da CollabNet e VersionOne (2019), é a medição do sucesso. Cerca de 52% dos indivíduos mede o sucesso através da satisfação do cliente ou utilizador, 48% através da criação de valor para o negócio, e 41% através da entrega no tempo certo.

B. *Agile* na gestão de projetos

O *agile* tem vindo a ser amplamente utilizado na gestão de projetos, onde é utilizada com sucesso não só em projetos de TI, mas também noutros contextos (Gustavsson, 2016).

Os princípios inerentes à gestão de projetos ágeis advêm dos valores e princípios definidos no Manifesto *Agile*, que apresentam grande ênfase nas pessoas, e na necessidade de flexibilidade e adaptabilidade, face à incerteza e complexidade presente na maior parte dos projetos (Fernandez e Fernandez, 2008).

Assim sendo, existe um conjunto de princípios e valores que caraterizam o *agile* na gestão de projetos, que são (Fernandez e Fernandez, 2008): assumir a simplicidade; abraçar a mudança; permitir e focar no próximo esforço; mudar incrementalmente; maximizar o valor; gerir o propósito e questionar as ações; o gestor de projetos deve gerir o projeto e as suas fronteiras; *feedback* rápido de todos os *stakeholders*; *deliverables* de qualidade; e, por fim, criação de documentação baseada no seu valor.

Destacam-se, no entanto, três pontos chave (Layton e Ostermiller, 2017): assegurar que a equipa é produtiva, e que pode, de forma sustentável, aumentar a sua produtividade ao longo do tempo; assegurar que a informação sobre o projeto e o seu progresso está disponível a todos os seus *stakeholders,* sem interromper o fluxo das atividades; e, por fim, responder à mudança e integrá-la no ciclo de desenvolvimento do produto. Estes princípios, valores e caraterísticas são, cada vez mais, representados em referenciais, dos quais são exemplos:

- Prince2 *Agile*

O Prince2 *Agile* integra os princípios, valores, comportamentos, técnicas, e conceitos do *agile* na camada de entrega, combinando estas caraterísticas com a base do Prince2, isto é, os seus princípios, temas, processos, papéis e artefactos (Cooke, 2016), sendo apenas aplicável a projetos, e não ao trabalho realizado diariamente (Axelos, 2018).

O termo "*agile"* é utilizado na ótica de um *mindset*, que traduz um conjunto de comportamentos, conceitos, *frameworks,* e técnicas amplamente utilizadas dentro da comunidade *agile*, como sendo parte de um método de trabalho (Axelos, 2018).

Posto isto, podem identificar-se os cinco princípios ágeis transpostos neste referencial, que são (Cooke, 2016): Transparência: assegurar que a informação sobre o projeto esta disponível e percetível a qualquer *stakeholder* do projeto (tais como a equipa de entrega, gestor de projetos, utilizadores do negócio, etc.); Colaboração: incentivar as equipas a trabalharem em conjunto através de uma visão partilhada, a representarem um papel ativo na tomada de decisão sobre os projetos, e, também, a assumir responsabilidade pelos resultados obtidos; Comunicação: utilizar ferramentas de comunicação que aumentem a transparência da informação, e suportem o trabalho colaborativo; Exploração: incentivar as equipas a explorar novas soluções para os problemas identificados, sendo o objetivo otimizar a utilização das competências de uma equipa, e reduzir o risco o mais cedo possível; Auto-organização: incentivar a equipa a distribuir o trabalho da melhor forma, tendo em conta as suas caraterísticas; Responsabilidade: criar uma visão e responsabilidade partilhada sobre o projeto, estimulando o sentido de pertença e propriedade partilhada.

O Prince2 *Agile* representa, então, um referencial híbrido de gestão de projetos, uma vez que combina o Prince2 clássico com o *mindset* ágil. A combinação é mais evidente em três áreas específicas (Cooke, 2016): criação de valor para o negócio, gestão de variáveis do projeto, e comunicação e apresentação de relatórios.

a. Criação de valor para o negócio:

No Prince2, o valor do projeto é identificado no seu início, sendo o valor percecionado que origina (ou não) a execução do projeto, sendo depois acompanhado e controlado. O *agile*, por sua vez, é

orientado à entrega de valor, onde primeiro são trabalhados os elementos do *backlog* que representam um maior valor para o cliente, sendo isto definido unicamente pelo *product owner*.

O Prince2 *Agile* combina estas duas perspetivas: o valor identificado na iniciação do projeto é utilizado como base para definir o produto, que alimenta diretamente o *product backlog* priorizado. Para além do envolvimento do *product owner* (que pode ser um ou mais, consoante a dimensão do projeto), considera também a presença de um executivo, um elemento sénior com autoridade de tomada de decisão sobre o produto, um ou vários representantes do negócio, e, também, de analistas do negócio e engenheiros de requisitos.

b. A gestão de variáveis de projeto:

Na gestão de projetos tradicional tende a existir um foco na gestão do tempo, custo e âmbito. No Prince2 *Agile* existe um conjunto de variáveis que são analisadas e classificadas em três categorias (não flexível, pode ser flexível ou flexível): O tempo e o custo têm valores fixos, que não podem ser alterados ao longo do projeto; O âmbito e a qualidade são flexíveis e existe uma grande probabilidade de variarem ao longo do projeto; e os benefícios e os riscos podem ser flexíveis e podem ser ajustados à medida que o projeto evolui.

No Prince2 *Agile*, a gestão do cronograma e do orçamento é realizada através da definição *a priori* de um conjunto fixo de valores e do estabelecimento de tolerâncias para possíveis exceções. É mais focado na entrega de valor através dos resultados obtidos, produzidos num tempo e custo fixos, e por uma equipa específica. Isto requer a obtenção de *feedback* e atualizações sobre a prioridade dos elementos do *backlog* e, também, sobre os resultados entregues em cada iteração.

c. Comunicação e apresentação de relatórios:

No Prince2 *Agile* a comunicação é constante entre os diversos *stakeholders* de um projeto. Para isso, realizam-se atualizações frequentes sobre o estado do projeto, as suas limitações e riscos.

- *Project Management Body of Knowledge* (PMBOK)

O PMBOK é um referencial de gestão de projetos que traduz uma orientação, tipicamente, em *waterfall*. No entanto, contempla, na atualidade, um apêndice dedicado à explicação dos ambientes de projeto ágeis, iterativos, adaptativos e híbridos (PMI, 2017), no qual é realizada uma organização dos diferentes ambientes de projetos num *continuum*, representado na Figura 4.

Adaptado de: (PMI, 2017)

Figura 4: *Continuum* dos ciclos de vida dos projetos

Aqui, contrariamente ao Prince2 *Agile*, o *agile* é apresentado como um ciclo de vida que pode ser atribuído a um projeto, e não como um *mindset*.

Este ciclo de vida carateriza-se pelo levantamento iterativo e frequente de requisitos, pela entrega mais frequente de pequenos incrementos do produto priorizados pelo cliente, pela incorporação da mudança em tempo real, pelo envolvimento constante dos *stakeholders* chave, e pelo controlo do risco e do custo consoante novos requisitos e riscos emergem (PMI, 2017).

Distinguem-se, ainda, duas formas de gerir e organizar estes ciclos de vida de projetos: iterações sequenciais compostas por um conjunto de fases; ou fases contínuas que se sobrepõem, coexistem e são iterativas.

- PM² *Project Management Methodology*

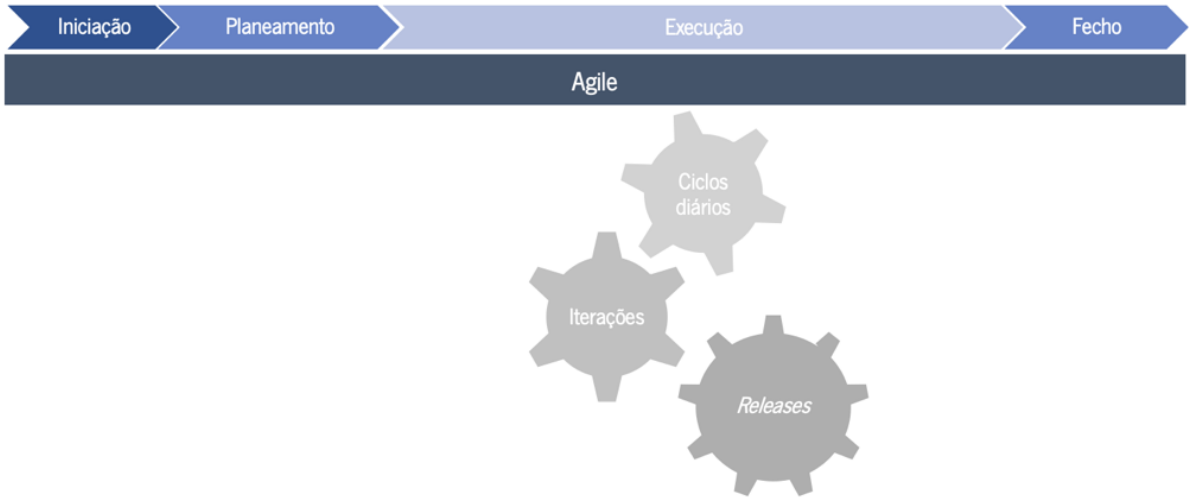
A PM² é uma metodologia de gestão de projetos desenvolvida pela Comissão Europeia, adequada à gestão de todo o ciclo de vida do projeto e a projetos de qualquer organização (EU, 2016).

Esta metodologia é suportada por quatro pilares: modelo de governança do projeto (conjunto de papéis e responsabilidades), ciclo de vida do projeto (fases de um projeto), conjunto de processos (atividades de gestão de projetos), e, por fim, um conjunto de artefactos (*templates* de documentação e linhas orientadoras). Considera também as boas práticas de gestão de projetos, e um conjunto de *mindsets*, que procuram consolidar os quatro pilares com as crenças e valores estabelecidos (EU, 2016).

A PM² tem associado um ciclo de vida *waterfall*, que se traduz nas suas práticas de gestão de projetos. No entanto, similarmente ao PMBOK, possui também um Apêndice onde contempla o *agile*.

Nesse apêndice, é disponibilizada uma estrutura que contempla as boas práticas do *agile*, enquanto continua com contratos rigorosos, auditorias de requisitos, boa coordenação ao nível de programas e portefólios, e colaboração com outros projetos, outras unidades organizacionais ou organizações externas (EU, 2016). Esta extensão é aplicável a projetos de TI, definindo os papéis e responsabilidades no *agile*, a integração com o ciclo de vida do projeto definido na PM², e, também, um conjunto de artefactos aplicáveis à extensão.

Na extensão, o ciclo de vida do projeto, representado na Figura 5, considera a realização de iterações na fase de execução, que correspondem à entrega de um incremento. É também prevista a realização de reuniões diárias e a possibilidade de entrega em *releases*.



Adaptado de: (EU, 2016)

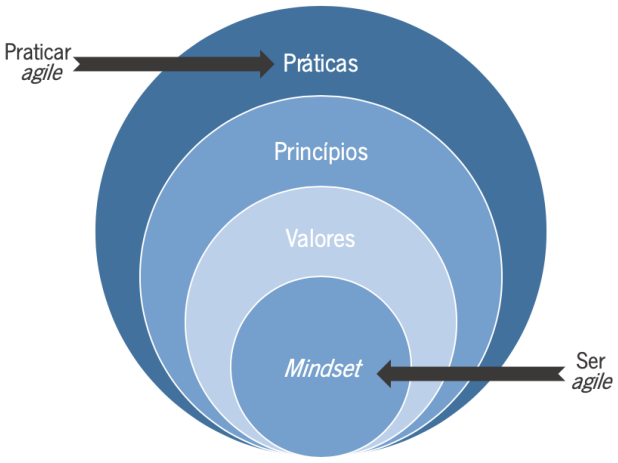
Figura 5: *Agile* no PM²

C. *Agile* como *mindset*

O *agile* ou a agilidade são muito mais do que um processo ou um conjunto de práticas, representando um *mindset* (Smith e Sidky, 2009).

Um *mindset*, por sua vez, consiste num conjunto de atributos e comportamentos que auxiliam as equipas dos projetos a atingir os objetivos identificados (Agile Alliance e PMI, 2017). É, então, uma forma de pensar sobre os projetos e as suas atividades, que pode ser aplicado a qualquer processo ou organização através de um conjunto de práticas (Smith e Sidky, 2009), ou seja, carateriza-se por um conjunto de valores, é guiado por princípios, e aplicado através de diferentes práticas (Agile Alliance e PMI, 2017).

Assim sendo, pode distinguir-se ser *agile* de praticar *agile*, conforme representado na Figura 6. Ser *agile*, consiste na adoção do *mindset* e dos valores e princípios que o definem, e na sua adaptação às diferentes situações que vão surgindo, enquanto praticar o *agile* consiste na aprendizagem e aplicação das práticas sem conhecer o *mindset* (Sidky, 2015).



Adaptado de: (Smith e Sidky, 2009)

Figura 6: Ser *agile* versus praticar *agile*

As práticas são úteis, uma vez que podem auxiliar a resolver problemas e a otimizar o fluxo de trabalho, no entanto é possível estabelecer um fluxo de trabalho sem este conjunto de práticas. Em contrapartida, o *mindset* ágil é visto como uma necessidade para a sobrevivência das organizações (Sidky, 2015), uma vez que representa a capacidade de criar e responder à mudança, e de equilibrar a flexibilidade com a estabilidade (Highsmith e Highsmith, 2002).

Posto isto, o *mindset agile* carateriza-se por (Forte e Kloppenborg, 2017; Sidky, 2015; Stellman e Greene, 2014): foco no valor para o cliente (satisfazer os clientes através da entrega de um resultado que satisfaz as suas necessidades), envolvimento dos *stakeholders* do projeto (envolver todos os *stakeholders* do projeto, incentivando à cooperação, partilha, e comunicação contínua), simplificar (manter a simplicidade a um ritmo sustentável, com ênfase na melhoria de processos), abertura à mudança, melhoria contínua, obtenção contínua de *feedback*, entrega contínua e incremental, e abertura e transparência sobre o projeto.

Cockburn (2016) classifica também o *mindset* ágil através de quatro termos: colaboração, entrega, reflexão e melhoria.

2.6. Implantação de Tecnologias da Informação (TI)

Tanto na literatura, como na vida quotidiana das organizações, é ouvido e utilizado com frequência o termo "implementação". No entanto, mostra-se importante distinguir os diferentes contextos de implementação e, por isso, emerge um novo termo: implantação.

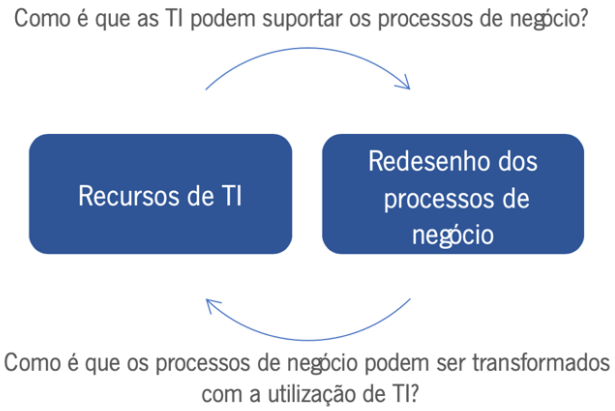
Numa primeira instância, estes dois termos parecem similares, no entanto, distinguem-se tendo em conta a sua aplicabilidade. D'Ascenção (2001) distingue estes dois termos, afirmando que implantação consiste numa fase de introdução ou alteração de processos e/ou sistemas existentes numa organização, traduzindo um processo com caraterísticas técnicas e organizacionais. Já implementação está relacionado com executar o que foi implantado (preparar para o funcionamento do novo processo).

Reascos e Carvalho (2019) distinguem também estes dois termos. Consideram que o termo implementação é ambíguo e utilizado de forma indiferenciada para diferentes cenários. Assim, caraterizam implantação como o processo que envolve a procura, avaliação, seleção, contratualização, e implantação de aplicação informáticas empresariais.

Davenport e Short (1990) relacionam as Tecnologias da Informação (TI) com o redesenho dos processos de negócio, afirmando que se trata de um processo recursivo, conforme representado na Figura 7, e que estes dois elementos são a chave um do outro. Nesta abordagem, os autores representam a capacidade das TI para mudar o negócio (foco tecnológico), e em como é que os processos de um negócio (novos ou já existentes) podem ser suportados pelas TI (foco organizacional).

Silveira e Diniz (2002) distinguem também o foco organizacional e tecnológico, no entanto, apresentam uma definição diferente desses termos e de como estes se relacionam. Para estes autores, o foco tecnológico consiste no uso de TI para impulsionar a mudança na organização, desencadeada pela implantação de um SI. Já o foco organizacional, consiste no foco na reestruturação de processos e na consciencialização de pessoas, colocando o SI num plano secundário.

Assim sendo, pode definir-se implantação como um processo sociotécnico para melhorar a organização, através do redesenho dos processos de negócio, numa perspetiva de desenvolvimento organizacional. Existe, portanto, um foco tecnológico, organizacional e social, onde a mudança é desencadeada por uma necessidade organizacional e, consequentemente, pela implantação do SI.



Adaptado de: (Davenport e Short, 1990)

Figura 7: Relacionamento entre TI e o negócio

A mudança organizacional pode ser definida, segundo Carvalho e Brito (2017), como uma alteração da estrutura e da forma de funcionamento de uma organização, sendo o objetivo aumentar a sua competitividade face ao seu ambiente externo. Um dos aspetos com maior relevância num processo de mudança organizacional é a gestão de expectativas, anseios e angústias dos colaboradores, sendo isto fundamental para se alcançar o sucesso da mudança e consequente adaptação (Silva e Neto, 2019). Assim, mostra-se importante introduzir dois termos essenciais que devem estar presentes em qualquer processo de implantação: gestão da mudança na organização, e gestão do sucesso do projeto e dos *deliverables*.

Apesar da sua designação, a gestão da mudança não consiste em gerir a mudança, mas sim em gerir as pessoas que estão a viver a mudança, e o impacto que uma mudança na organização ou no seu ambiente pode representar (Moran e Brightman, 2000). Consiste, portanto, na avaliação dos aspetos humanos das organizações, relacionados, por exemplo, com a gestão de emoções e reações humanas (Maurer, 2001; Machado e Neiva, 2017).

Assim, aquando a realização de uma mudança, deve procurar-se envolver as pessoas afetadas pelo processo de mudança e criar um ambiente onde estes possam discutir e perceber as novas ideias e alinhar-se com a nova estratégia e/ou processo a implantar (Moran e Brightman, 2000).

Já no que respeita à gestão de sucesso, esta deve, segundo Varajão (2016), ser suportada por um processo que contemple:

- Gestão do sucesso do projeto: atividade focada no processo de gestão do projeto e na sua realização bem-sucedida, em termos de custo, tempo e âmbito;
- Gestão do sucesso dos *deliverables* do projeto: está relacionado com o impacto dos produtos e/ou serviços resultantes do projeto.

2.6.1. Métodos de implantação de TI

Um método de implantação consiste num conjunto de elementos que servem de guia às diferentes equipas aquando da implantação de uma aplicação informática ou TI (Gulledge e Simon, 2005), sendo um bom exemplo o caso dos *Enterprise Resource Planning* (ERP).

Existem diversos *frameworks*, cujo objetivo é orientar o processo de implantação, no entanto, segundo Reascos e Carvalho (2018), todos eles cobrem um problema específico, o que se mostra uma desvantagem perante a necessidade de uma visão global do processo.

Os mesmos autores realizaram uma análise, onde identificaram seis tipos de *frameworks* existentes, que são: para apoio à decisão, para a etapa de pré-implantação, para seleção de *software*, baseadas na *framework* TOE (Tecnologia, Organização e Ambiente), para adoção na *cloud,* e, por fim, *frameworks* holísticos.

Posto isto, pode dizer-se que existem várias formas de implantar uma TI numa organização. Caso uma organização não possua uma TI, o processo é mais simples, ou seja, trata "apenas" da implantação e da integração dos processos organizacionais. No entanto, aquando a existência de uma aplicação informática a situação é mais complexa.

Shelly et al. (2007), conforme representado na Figura 8, identificam quatro formas possíveis para realizar este processo:

- **Corte direto**: as duas TI não coexistem, apenas prevalece a nova aplicação;
- **Existência em paralelo**: as duas TI coexistem totalmente funcionais durante um período específico de tempo;
- **Piloto**: consiste na implantação completa da nova TI, apenas numa parte da organização, a título experimental;
- **Faseado**: a implantação da nova aplicação é feita por etapas ou por módulos, coexistindo as duas aplicações.

Adaptado de: (Shelly et al., 2007)

Figura 8: Formas de realizar o processo de implantação

A. Classificação dos métodos de implantação de TI

Vários são os métodos de implantação de TI desenvolvidos pelas organizações para a implantação dos seus produtos, com o objetivo de otimizar a entrega, aumentar a competitividade, e aumentar a satisfação dos seus clientes. São também estes os motivos que eventualmente levam as organizações a não partilhar os seus métodos de implantação abertamente, por estarem associados ao ganho de vantagens competitivas.

No âmbito desta dissertação, foram identificados 19 métodos de implantação: *SAP Activate Methodology*, *Odoo Implementation Methodology*, *Expertise@Work*, *ServiceNow Implementation Methodology*, *Microsoft Sure Step*, *ASI Client Success Agile Implementation Methodology*, *SunSystems Implementation Method*, *Kronos Paragon Implementation Methodology*, *Oracle Unified Process*, *Infor Deployment Method*, *VersionOne Implementation Services*, *Cognos Solutions Implementation Methodology*, *IFS Implementation Methodology*, *Ideal Implementation Methodology*, *PeopleSoft Implementation*, *IBM Business Analytics Solutions Implementation Method*, *Accenture CAS: Solution Implementation*, *Fast Track Cloud Implementation Methodology*, e Metodologia de Implementação Primavera.

Os métodos de implantação referidos foram classificados em quatro categorias: *Waterfall* (W), *Agile* (A), Iterativo e/ou Incremental (I), e Híbrido (H). Para a classificação, foram utilizadas as caraterísticas identificadas nas secções 2.1 a 2.5 deste trabalho de dissertação. Os critérios identificados encontram-se representados na Tabela 13.

Tabela 13: Critérios de classificação dos métodos de implantação

	Waterfall	*Agile*	Iterativo	Incremental
Entrega	Uma	Frequente, incremental	Frequente	Frequente

	Waterfall	*Agile*	Iterativo	Incremental
			(cada iteração acrescenta valor à anterior)	(cada entrega é um incremento totalmente completo)
Requisitos	Definido a *priori*	Progressivamente identificados e atualizados	Em intervalos periódicos	Em intervalos periódicos
Planeamento	*A priori*	Em cada iteração	O planeamento da próxima iteração é realizado conforme o trabalho progride	-
Envolvimento do cliente	Muito baixo	Envolvido frequentemente e continuamente	Envolvido regularmente	Envolvido regularmente
Feedback do cliente	Em *milestones* específicas	Frequente	Frequente	-
Processo	Sequencial	Iterativo e incremental	Iterativo	Incremental
Âmbito do projeto	Controlado através de um planeamento detalhado, elaborado nas primeiras fases do ciclo de vida do projeto	Processo contínuo, gerido em cada iteração	Globalmente determinado cedo no projeto	-
Mudança	Evita ao máximo	Incorporada em tempo-real e resolvida rapidamente	Em intervalos periódicos; O tempo e o custo são constantemente modificados.	Em intervalos periódicos

Posto isto, a classificação atribuída encontra-se representada na Tabela 14.

Tabela 14: Classificação atribuída aos métodos de implantação

	Empresa	Métodos de implantação	Classificação
1	SAP SE	*SAP Activate Methodology*	A
2	Odoo	*Odoo Implementation Methodology*	A
3	SABA	*Expertise@Work*	A
4	ServiceNow	*ServiceNow Implementation Methodology*	H
5	Microsoft	*Microsoft Surestep*	H
6	Advanced Solutions International	*ASI Client Success Agile Implementation Methodology*	H
7	LLP Group	*SunSystems Implementation Method*	H
8	Kronos	*Kronos Paragon Implementation Methodology*	H
9	Oracle	*Oracle Unified Process*	I
10	Infor	*Infor Deployment Method*	I

	Empresa	Métodos de implantação	Classificação
11	Collab Net	*VersionOne Implementation Services*	I
12	IBM	*Cognos Solutions Implementation Methodology*	W
13	IFS	*IFS Implementation Methodology*	W
14	SysPro	*Ideal Implementation Methodology*	W
15	Graycell	*PeopleSoft Implementation*	W
16	IBM	*IBM Business Analytics Solutions Implementation Method*	W
17	Accenture	*Accenture CAS: Solution Implementation*	W
18	Cherry Road Technologies	*Fast Track Cloud Implementation Methodology*	W
19	Primavera BSS	Metodologia de Implementação Primavera	W

Legenda: W – *Waterfall* (preditivo), A – *Agile*, I – Iterativo e/ou incremental

B. Métodos ágeis de implantação de TI

Depois de classificados os métodos de implantação de TI, apresentados na Tabela 14, revelou-se importante realizar uma análise sobre os métodos ágeis de implantação de TI.

1 *SAP Activate Methodology*

A SAP *Activate Methodology* foi desenvolvida, conforme indicado pela sua designação, pela SAP *Software Solutions,* em 2015. Marca uma evolução no paradigma de implantação do ERP da SAP, que abandonou uma perspetiva de consultoria, onde o cliente decidia o rumo do processo de implantação, para uma perspetiva onde a SAP guia o cliente através de uma abordagem ágil, com entregas mais rápidas (Zamfir e Paul, 2017).

Este método de implantação contempla seis etapas principais (SAP, 2018) (Zamfir e Paul, 2017):

a. Descoberta (*Discover*): desenvolvimento das estratégias e *roadmap* a utilizar no projeto de implantação. Nesta etapa o cliente tem também a possibilidade de experimentar o produto (na sua versão *standard*), antes do projeto de implantação se iniciar.
b. Preparação (*Prepare*): iniciação do projeto. Nesta etapa são finalizados os planos e é atribuída uma equipa ao projeto.
c. Exploração (*Explore*): verificar que os diferentes cenários da aplicação informática vão de encontro às necessidades do negócio. Aqui decorre uma análise *fit/gap*, com o objetivo de se validar a aplicação informática e preparar o plano das *sprints* e *releases*. Procura identificar o âmbito do projeto.
d. Realização (*Realize*): esta etapa foca-se na transferência dos requisitos de negócio para uma parametrização da aplicação informática (ou de uma parte dela). Esta atividade inclui não só a configuração, mas também a realização de integrações e migrações de dados.

e. Implantação (*Deploy*): preparar o ambiente de produção, conduzir a transição e alterar as operações de negócio. Aqui há um *cutover* das funcionalidades em análise e passa a ser utilizado, em produção, o incremento trabalhado na *sprint*.

f. Execução (*Run*): ocorre no final da realização de todas as iterações e consiste na otimização das operações.

Cada iteração é composta pelas etapas de preparação, exploração, realização e implantação, fazendo com que a entrega seja incremental.

Por fim, mostra-se também importante realçar que o SAP Activate se aplica a todos os produtos SAP, incluindo em *cloud*, *on-premise* e híbridos (SAP, 2018), tendo substituído o SAP ASAP (aplicado aos produtos *on-premise*) e o SAP *Launch* (aplicado aos produtos *cloud*).

2 *Odoo Implementation Methodology* (ODM)

A ODM tem a sua aplicabilidade na implantação do ERP *open source* Odoo, sendo o seu objetivo disponibilizar o produto no menor tempo e custo possível (Odoo, s.d.).

O processo de implantação é composto por três elementos (Odoo, s.d):

1. *Kick-off*: esta etapa consiste numa reunião de *kick-off*, onde é elaborado e proposto um plano de implantação. O objetivo é perceber o negócio do cliente.
2. Implantação: a implantação é realizada através de iterações (chamadas fases no ODM). Cada iteração pode constituir um ou mais módulos, dependo do projeto, e é, sempre, composta pelas seguintes etapas:
   - *On-boarding*: consiste na revisão, validação e configuração dos processos de negócio com o gestor de projetos da Odoo e o cliente;
   - Dados: importação de dados de uma aplicação já existente;
   - Formação: formação dos utilizadores, verificação da qualidade e customização;
   - Produção: transição para ambiente de produção e iniciação do uso do incremento neste ambiente.
3. Customizações: avaliação e refinamento de processos de negócio e da aplicação informática. Podem, por exemplo, ser atividades de melhoria e/ou automatização.

Por fim, inicia-se a etapa de suporte, que ocorre depois da etapa de implantação (no pós-projeto). Serve para clarificar dúvidas sobre a aplicação informática, resolver problemas que possam surgir, e apoiar o cliente na fase de pós-projeto (Odoo, s.d).

Este é um bom exemplo de um método ágil de implantação. Algumas caraterísticas importantes são a existência de ciclos iterativos e incrementais, a transparência, o envolvimento frequente do cliente no

projeto, comunicação frequente, maior *feedback* e colaboração do cliente, e, também, o maior foco e ajustamento ao cliente, ao seu negócio, necessidades e especificações.

No entanto, considera-se que o suporte deveria estar presente deste a primeira disponibilização do produto ao cliente, e que deveria estar claro como se avalia e responde à mudança (de requisitos ou necessidades, por exemplo).

3 *Expertise@Work*

O Expertise@Work é utilizado para a implantação de uma aplicação informática *cloud* para gestão de formações, desenvolvida pela SABA (2014). Segue uma abordagem por *sprints* e é composto por quatro etapas (SABA, 2014): descoberta (*discover*), configuração (*configure*), capacitação (*enable*), e validação (*validation*).

Cada *sprint* é composta pelas quatro etapas apresentadas e tem uma duração curta, de quatro a seis semanas, resultando, no final, um incremento funcional, pronto a utilizar em ambiente de produção (SABA, 2014). É, também, importante realçar que existe uma priorização dos incrementos disponibilizados, sendo primeiro trabalhado e entregue a parte do produto que o cliente mais precisa e/ou deseja (SABA, 2014).

É também utilizado por parceiros da SABA (SABA, 2014), que comercializam esta aplicação informática. Os parceiros podem moldar o Expertise@Work às suas necessidades e às dos seus clientes. Assim, na versão utilizada por alguns parceiros, existem duas fases que precedem a realização de *sprints*: teste tecnológico (demonstração da aplicação informática) e prova de conceito (validação do produto através da sua disponibilização e teste de cenários de negócio, realizada também por *sprints*).

Outra diferença face à versão original da SABA (2014) é a forma de entrega, uma vez que os incrementos disponibilizados em cada *sprint*, na fase de implantação, não são utilizados em ambiente de produção, ou seja, existe apenas um *go-live,* no final do projeto, o que tipicamente acontece nos métodos com uma orientação em *waterfall*.

2.7. *Method engineering*

A engenharia de métodos (*Method Engineering ou Methodology Engineering*) tem a sua origem na engenharia mecânica, onde foi definida como uma abordagem sistemática para melhoria de métodos de trabalho (Harmsen et al., 1994).

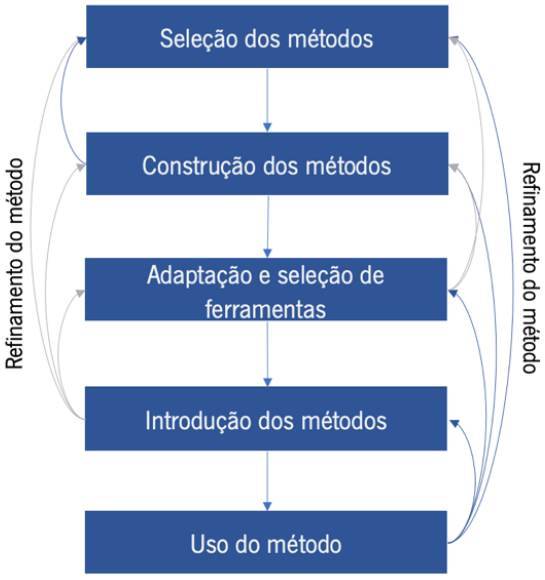
Não obstante ao apresentado acima, mas num tom mais genérico, pode dizer-se que a Engenharia de Métodos (ME) consiste numa área de engenharia responsável pelo *design,* construção, e adaptação dos métodos, técnicas e ferramentas para o DSI (Brinkkemper, 1996).

Assim, o seu foco não se encontra na aquisição de métodos *standard,* mas sim na construção de um método à medida de cada organização (Henderson-Sellers, 2006). Consiste, assim, numa abordagem sistemática e coordenada para o estabelecimento de métodos (Odell, 1996)

Tolvanen (1998) reconhece que as organizações tendem a desenvolver os seus próprios métodos ou a adaptar um método já existente à sua realidade. O mesmo autor, conforme representado na Figura 9, identifica cinco etapas pelas quais as organizações passam no desenvolvimento interno de métodos:

1. Seleção de métodos: decisão sobre quais os métodos a seguir ou utilizar,
2. Construção do método: consiste na construção, melhoria ou modificação de um método, tendo em conta os métodos selecionados,
3. Seleção e adaptação de ferramentas: representação e implementação do novo método numa ferramenta de apoio/suporte ao método. Esta não é uma etapa obrigatória.
4. Introdução do método: consiste na iniciação da utilização do método, incluindo atividades de formação e criação de projetos piloto,
5. Uso do método: utilização do método criado em conjunto com as ferramentas de suporte.

Na literatura existe, segundo Ralyté et al. (2004), uma grande variedade de abordagens à ME, que apoiam a criação de novos métodos, a adaptação de métodos já existentes a mudanças (que ocorreram ou que são necessárias) e, também, a adaptação a situações especificas de um projeto.



Adaptado de: (Tolvanen, 1998)

Figura 9: Etapas para o desenvolvimento de métodos

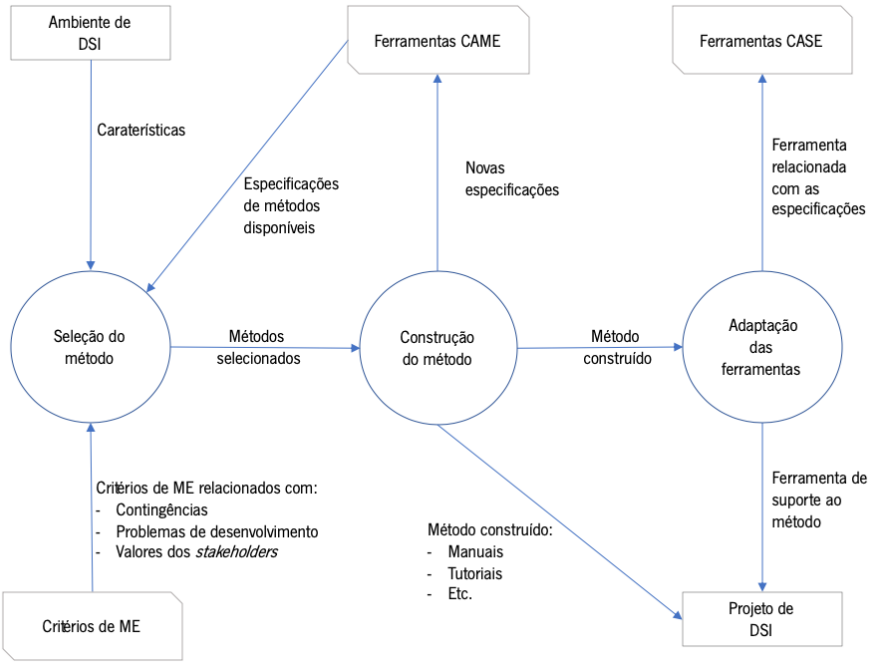
Genericamente, Tolvanen (1998) distingue duas abordagens ao ME:

1. **Baseada na organização:** o processo de desenvolvimento é similar em toda a organização e o método é apropriado a todos os projetos;

2. **Baseada no projeto:** os métodos são desenvolvidos baseados em cada projeto, dado o processo de desenvolvimento ser diferente em cada caso.

Ralyté et al. (2004) descrevem também quatro tipos de abordagens à ME, que não são mutuamente exclusivas às abordagens identificadas por Tolvanen (1998). As abordagens são as seguintes: ***Ad-hoc*** (o novo método é desenvolvido de raiz), **baseada em paradigmas** (é utilizado um modelo ou meta modelo como base de análise, representativo da situação AS-IS. Este modelo será instanciado e adaptado, sendo o resultado final o desenvolvimento do modelo TO-BE), **baseada em extensões** (propõe melhorias (através da introdução de novos conceitos e propriedades) a um método já existente) e **baseado em componentes** (propõe a reutilização de partes de outros métodos, na criação de um novo método ou melhoria de um método já existente).

Na Figura 10 encontra-se o processo identificado por Tolvanen (1998), onde podem ser identificados quatro elementos principais: etapas de ME (círculos), entidades externas (retângulos), armazenamento de dados (retângulos arredondados/cortados) e setas (fluxos de dados entre os diferentes elementos).



Adaptado de: (Tolvanen, 1998)

Figura 10: Processo de *Method Engineering*

Conforme pode ser visualizado na Figura 10, o processo de ME é composto por três etapas:

1. **Seleção do método:** o ambiente de desenvolvimento do SI é analisado, tendo em conta os critérios de ME. Estes critérios podem ser independentes da situação (por exemplo, facilidade de uso do método) ou dependentes da situação (traduzem aspetos relevantes de outros métodos que devem ser tidos em conta para a satisfação dos objetivos do novo método). De uma forma geral, os critérios

podem ser organizados em três categorias: contingências, problemas de desenvolvimento e valores dos *stakeholders*.

2. **Construção do método:** após seleção do(s) método(s), o novo método será construído, tendo em conta novos componentes. Para que a construção se realize é necessário a utilização de *metamodeling* (desenvolvimento de um modelo do método).
3. **Adaptação de ferramentas:** consiste na adoção, adaptação ou desenvolvimento de uma ferramenta para o método, com o objetivo de o descrever detalhadamente e, com isso, aumentar a probabilidade de adesão e uso ao novo método. Ainda nesta etapa devem ser elaborados manuais, tutoriais, entre outros elementos que facilitem e suportem a utilização e introdução do novo método.

Tolvanen (1998) não inclui, no entanto, algumas etapas de ME na Figura 10, tais como a introdução ao método, o uso do método, e a coleção de experiências; reconhece, no entanto, a sua existência.

2.8. Trabalhos relacionados

Neste capítulo são abordados outros trabalhos cujo foco se relaciona com o tema desta dissertação, que é o processo de construção de um método de implantação de Tecnologias da Informação (TI).

O foco desta dissertação é o desenvolvimento de um novo método de implantação de TI, que incorpore os princípios, valores e caraterísticas dos ciclos de vida ágeis e da Primavera BSS. Este é um trabalho habitualmente realizado dentro das organizações, que tendem a não partilhar o processo de desenvolvimento e/ou adaptação dos seus métodos, por o associarem à obtenção de vantagens competitivas. Por isso, o número de estudos relevantes e com âmbito similar a este trabalho é bastante reduzido na literatura.

2.8.1. *SAP Activate Methodology* (SAP, 2018)

O processo de transição realizado pela SAP é um bom exemplo da natureza do presente trabalho: a transição de uma abordagem clássica (*Accelerated* SAP e SAP Launch) para uma abordagem ágil (SAP *Activate Methodology*), desencadeado pelo lançamento de um novo produto hospedado na *cloud*. Assim, o SAP Activate reúne, então, os princípios ágeis e os requisitos e necessidades associadas à *cloud*.

No entanto, apesar de se encontrarem disponíveis ambos os métodos, a SAP não disponibiliza o processo de trabalho realizado até se chegar ao produto final. Afirmam apenas que combina três pilares (SAP *Guided configuration*, SAP *Best Practices* e SAP *Methodology*), e que é utilizado nas modalidades *on-premise*, *cloud* e híbrido.

2.8.2. *SImplE: Framework* para implantação de aplicações empresariais (Reascos e Carvalho, 2018; Reascos e Carvalho, 2019)

Neste trabalho foi desenvolvido um modelo de processo para a implantação de aplicações informáticas empresariais, adequado a pequenas e médias empresas. Globalmente, o modelo distingue três fases do processo de implantação (Reascos e Carvalho, 2018; Reascos e Carvalho, 2019):

A. Pré-implantação: realizada pela organização e consiste no estabelecimento da visão do projeto, definição de processos a automatizar, identificação de requisitos da aplicação, seleção da aplicação informática, negociação com o fornecedor e, por fim, tomada de decisão.
B. Implantação: realizada pela organização e pelo fornecedor de *software*. Aqui, o fornecedor irá conhecer a organização cliente, customizar a aplicação informática, migrar os dados, testar, realizar formação e, por fim, lançar a aplicação em ambiente de produção.
C. Pós-implantação: consiste na realização de suporte pelo fornecedor de *software*, sempre que solicitado pela empresa cliente.

São também acrescentados três elementos que decorrem desde a fase de pré-implantação à fase de pós-implantação (Reascos e Carvalho, 2019): liderança e comunicação, gestão da mudança, e gestão do projeto.

2.8.3. Método para implantação de ERP (Xu et al., 2010)

Este trabalho consistiu no desenvolvimento de um *roadmap* para a implantação de ERP em pequenas e médias empresas, que é composto pelas seguintes fases:

A. Avaliação da prontidão: ocorre na fase de pré-implantação e consiste na avaliação do estado da organização e da sua prontidão para a implantação do ERP. Para isso, são realizadas entrevistas aos *stakeholders* mais relevantes, analisam-se os resultados obtidos, identificam-se as principais preocupações, e é proposto um *roadmap* para o projeto.
B. Gestão dos processos de negócio: consiste no mapeamento dos principais processos e fluxos de negócio (AS-IS), na identificação dos *Key Performance Indicators* (KPI), dos principais requisitos, no desenho dos novos processos de negócio (TO-BE) e, finalmente, no planeamento da *framework* de TI.
C. Seleção do ERP: consiste na recolha de dados sobre o ERP, seleção preliminar da aplicação, teste das principais funcionalidades, avaliação final, tomada de decisão, e preparação para a implantação e *design* da aplicação.
D. Implantação: aqui, o objetivo é a configuração da aplicação informática para que esta suporte os processos de negócio desejados. As atividades desta etapa são a definição do âmbito do projeto,

formação dos utilizadores, modelação dos principais processos de negócio, configuração da aplicação informática, lançamento da aplicação e, por fim, refinamento do sistema e dos seus processos.

2.8.4. Implantação ágil de um ERP (Isetta e Sampietro, 2018)

O trabalho realizado neste artigo consistiu na avaliação da aplicabilidade do *agile* no processo de implantação de ERP. Os autores concluíram que o *agile* ainda não é comumente adotado na implantação de ERP. Todavia, identificam um conjunto de princípios e boas práticas para a sua adaptação ao contexto de implantação, que foram: maximizar o valor para os *stakeholders*, abraçar a mudança, entregar *software* funcional como objetivo primário, assumir a simplicidade, mudar incrementalmente, gerir com um propósito (que deve ser satisfazer os *stakeholders*), garantir a continuidade do projeto (melhoria contínua), rápidos ciclos de *feedback* e, por fim, realizar a manutenção dos incrementos entregues e assegurar que estes estão de acordo com o valor desejado para o negócio. Identificam também alguns desafios, tais como a gestão de dependências, resistência à mudança, período de adaptação à mudança e, também, o período de aprendizagem do novo método de trabalho.

A conclusão sobre a aplicabilidade do *agile* num projeto de implantação não é uma resposta linear e é totalmente dependente das caraterísticas de cada projeto (*upgrade versus* novo sistema, quantidade de customizações, confiança no cliente, competências da equipa e da organização cliente, etc.).

2.8.5. Método desenvolvido por Serour e Henderson-Sellers (2004)

Este trabalho consiste num caso de *Situational Method Engineering* (SME), cujo objetivo é o desenvolvimento de um método ou processo ágil específico para o caso particular de uma organização e dos seus projetos, que se focasse nas pessoas (e não nos processos), sendo simples, iterativo e incremental.

O primeiro passo consistiu em perceber e conhecer a organização e a sua necessidade de mudança. De seguida, os autores procederam à análise de diferentes métodos para o desenvolvimento de TI existentes no mercado, da cultura da organização (atual e desejada) e, por fim, ao desenvolvimento e aplicação do novo método.

Este é um método para o desenvolvimento de *software*, o que não é o caso desta dissertação. No entanto, este exemplo encontra-se nesta secção porque, em primeiro lugar, existe a construção de um método, e depois, porque é explicitado todo o processo de construção através de um estudo empírico realizado numa empresa internacional.

2.8.6. *Cork Organisational Standard Software Process* (Fitzgerald et al., 2000)

Este estudo, de Fitzgerald et. al (2000), descreve a adaptação de um processo de desenvolvimento de *software* a um caso real de uma organização, a Motorola. O processo, chamado *Cork Organisational Standard Software Proc*ess (OSSP), é bem documentado, tem um ciclo de melhoria contínua, e é adaptado para cada projeto.

O processo é dividido em três níveis:

A. Industrial: componentes de domínio público, o que significa que todas as organizações podem ter acesso. Neste nível a Motorola adotou o *Standard* IEEE 1074 (providencia um conjunto de atividades para o desenvolvimento e manutenção de *software*) e o modelo em V (explicita um conjunto de atividades de desenvolvimento sequenciais).
B. Organizacional: adaptação do *standard* adotado à realidade da organização.
C. Projeto: adaptação do método num nível micro, no caso, ao projeto e às suas necessidades operacionais.

A temática deste artigo assenta no desenvolvimento de *software*, o que não é, novamente, o foco deste trabalho de dissertação, contudo o seu conteúdo e a lógica associada tornam-no particularmente interessante e importante. Realça que, apesar de existirem diversos *standards* e documentos que descrevem processos e a sua aplicabilidade, as organizações devem analisar, pensar e planear os seus processos internos. Ou seja, não se deve adaptar uma organização a um método, mas o método a uma organização, dado que possuem caraterísticas únicas e intrínsecas que as distinguem das restantes.

2.8.7. *AgilIS (Varajão, 2018)*

O agilIS é apresentado como um modelo de processo ágil para o Desenvolvimento de Sistemas de Informação (Varajão, 2018). Para além de conferir agilidade ao processo de DSI, contempla a integração do desenvolvimento com a formação, gestão, planeamento e operações de Sistemas de Informação.

Surgiu no contexto académico para a finalidade de ensino. Atualmente, encontra-se em desenvolvimento e experimentação em contexto prático.

2.9. Reflexão crítica sobre o estado da arte

Nos últimos anos, tem-se assistido a uma mudança de paradigma, onde os ciclos de vida ágeis têm ganho relevância (Solinski e Petersen, 2016). Este tipo de ciclo de vida tem atraído cada vez mais a atenção das organizações (Mahalakshmi e Sundararajan, 2013), tanto pelos seus princípios, quanto pelas suas promessas de vantagens. Contudo, raramente são utilizadas na sua forma *standard* (Solinski e Petersen, 2016), isto é, são adaptados a um contexto organizacional, o que é um ponto positivo e uma correta interpretação sobre como os utilizar.

Relativamente à análise das forças e limitações dos diferentes modelos analisados, deve ser mencionado que existem poucas evidências empíricas que as confirmem (Begel e Nagappan, 2007; Laanti et al., 2011; Osorio et al., 2011; Wells, 2012; Kamei et al., 2017). Este é, no entanto, um aspeto importante, pelo que esta lacuna deve ser preenchida por trabalhos futuros. Com base na análise das forças e limitações, é possível identificar algumas das principais caraterísticas dos modelos analisados, conforme representado no Apêndice A.9, onde são descritos os diferentes modelos tendo em conta 27 caraterísticas que emergiram da análise elaborada.

O *agile* foi amplamente explorado no desenvolvimento de *software*, área à qual está tipicamente associado, sendo, inclusive, caraterizado pelos princípios e valores identificado no Manifesto *Agile* (Beck et al., 2001).

Os modelos ágeis não são adequados a equipas grandes e são difíceis de aplicar em grandes projetos. Aquando a utilização deste tipo de modelos, é sentida uma perda da visão global do projeto e uma necessidade de maior coordenação e comunicação. As equipas são auto-organizadas e apresentam elevada satisfação, no entanto, requer que permaneçam no mesmo espaço físico durante longos períodos de tempo, e dificulta a identificação dos contributos individuais. O trabalho é priorizado e a entrega ocorre através de pequenos incrementos, disponibilizados no final de cada iteração. É, também, incremental, iterativa e evolutiva, com um tempo de colocação no mercado reduzido. O produto entregue possui, potencialmente, maior qualidade, e de estar de acordo com as necessidades e expectativas do cliente, uma vez que este é significativamente envolvido no processo, fornece *feedback* constantemente, e está em comunicação direta com toda a equipa do projeto. Este tipo de modelo, carateriza-se, também, pela sua capacidade de resposta à mudança, de integrá-la imediatamente, e pela maior e mais rápida identificação e resolução de problemas e erros. No geral, a documentação existe na medida certa, existe uma maior visibilidade e transparência do projeto, maior interação, colaboração, partilha, comunicação, e maior facilidade na monitorização e controlo do projeto. Pode representar alguns custos (porque, caso não seja gerido adequadamente, pode transformar-se num projeto "infinito"), mas, globalmente, as equipas reportam uma redução de custos.

A aplicabilidade do *agile* já não é exclusiva às TI, incluindo situações que são de grande escala, de natureza complexa e que acontecem num elevado número de contextos (Axelos, 2018), dos quais são exemplos a consultoria, a gestão de projetos, a saúde, educação, manufatura, entre muitas outras (Agile Alliance e PMI, 2017). Assim sendo, pode ser analisado como um *mindset*, caraterizado pelo foco e envolvimento do cliente e dos restantes *stakeholders* durante todo o projeto, fornecendo *feedback* constante sobre o trabalho e respetivas necessidades. Carateriza-se, igualmente, pela abertura à mudança, melhoria contínua, reflexão, entrega contínua e incremental, grande abertura e visibilidade

sobre o projeto, e por um maior nível de colaboração (Forte e Kloppenborg, 2017; Sidki, 2015; Stellman e Greene, 2014; Cockburn, 2016).

O processo de implantação de TI é composto por duas dimensões: tecnológica e organizacional (Davenport e Short, 1990; D'Ascenção, 2001; Silveira e Diniz, 2002), cujo objetivo é melhorar a organização, os seus processos e o seu desempenho.

A utilização do *agile* no contexto de implantação de TI é ainda bastante limitada, correspondendo apenas a 16% dos métodos de implantação analisados. Apesar de o *agile* estar a ganhar relevância e a levantar a curiosidade das organizações, os métodos de implantação com ciclos de vida em *waterfall* continuam a ser os preferidos (46%), seguidos dos ciclos de vida híbridos (26%). No entanto, de acordo com o trabalho realizado, o *agile* é aplicável ao contexto de implantação de TI, sempre que analisado como um *mindset*, sendo esta a perspetiva a adotar nesta dissertação.

Posto isto, a principal mensagem a retirar do trabalho de revisão realizado é que não existe um modelo ou método "melhor", "apenas" existem modelos e métodos que são adequados para um determinado contexto com certas caraterísticas. A aplicabilidade do *agile*, e dos restantes ciclos de vida, não é uma decisão linear, mas sim uma decisão que está dependente das caraterísticas das organizações, dos projetos, e das diferentes equipas que os compõe. Então, deve ser efetuada uma análise detalhada destes elementos e, consequentemente, ser elaborado um plano para a sua adaptação, introdução e utilização, tendo em conta o contexto onde se irá inserir.

3 Plano de investigação

Nesta secção são descritos o problema, o método adotado para a realização da dissertação, o processo de investigação, e a estratégia de pesquisa utilizada.

3.1. Problema

O problema a resolver surge no contexto empresarial da *Primavera Business Software Solutions* (Primavera BSS), nomeadamente no que diz respeito aos métodos utilizados atualmente para a implantação dos seus produtos, descritos na Metodologia de Implementação Primavera (MIP).

A MIP é utilizada para a implantação da maior parte das aplicações existentes no universo Primavera. Tem subjacente uma abordagem em *waterfall*, onde a entrega é realizada em *big-bang*, a documentação é extensa e *standard,* e onde consta um conjunto predefinido de atividades a realizar sequencialmente. No entanto, existem também caraterísticas que lhe fornecem um caráter mais dinâmico, tal como a possibilidade de análise e introdução de mudanças num projeto (embora seja evitada).

Existe um conjunto de fatores que propulsionaram este trabalho e que o justificam:

- Dificuldade no fecho dos projetos de implantação: os clientes "não querem" fechar o projeto, devido à entrega *big bang* de uma nova e complexa TI, que ocorre apenas no final do projeto (através de um corte direto), juntamente com a formação dos utilizadores finais.
- Adequação ao novo ERP em lançamento: no contexto de inovação tecnológica, a Primavera BSS irá lançar um novo ERP nativo em *cloud*, o Rose, que, pelas suas caraterísticas, requer a revisão do processo de implantação dos produtos Primavera.
- Incorporação das boas práticas existentes no mercado: o *agile* é um tema emergente que encontra a sua aplicabilidade no desenvolvimento de *software*, onde os seus benefícios e limitações já se encontram comprovados. Começa, no entanto, a surgir a sua aplicabilidade noutras áreas, tais como na gestão de projetos, e no contexto de implantação de TI. Alguns exemplos são os referenciais Prince 2 *Agile*, a versão mais recente do PMBOK, e casos de empresas como a SAP e a Microsoft.
- Solicitação de uma abordagem ágil pelos clientes e parceiros: conforme referido, a MIP (composta por três métodos) é utilizada na atualidade para a implantação das aplicações do universo Primavera. Os métodos que a compõe (*accelerated, extended* e *controlled*) são métodos estruturados, com etapas e documentação predefinidas, e com uma entrega única prevista. Assim, cada vez mais, clientes e parceiros desejam uma maior agilidade para estas implantações, inspirados pelo emergir do *agile*, e pela sua aplicabilidade por algumas das maiores empresas internacionais no contexto de implantação.

É neste âmbito que surge o presente trabalho de dissertação, cujo objetivo é o desenvolvimento de um novo método ágil de implantação de TI (o AgileMIP), aplicável tanto a modalidades *on-premise*, como *cloud*. Este novo método visa permitir a exploração de todas potencialidades dos produtos Primavera e a sua correta implantação, proporcionando uma maior agilidade. Por esse motivo, este método procura suportar os processos e os métodos de trabalho inerentes às equipas Primavera, como também incorporar os princípios e valores ágeis. Posto isto, estabeleceram-se as seguintes questões de investigação:

- Questão 1: Quais as caraterísticas do *mindset* ágil no contexto de implantação de TI?
- Questão 2: Que caraterísticas deve reunir um método ágil para a implantação dos produtos Primavera?

3.2. Descrição e justificação do método de investigação

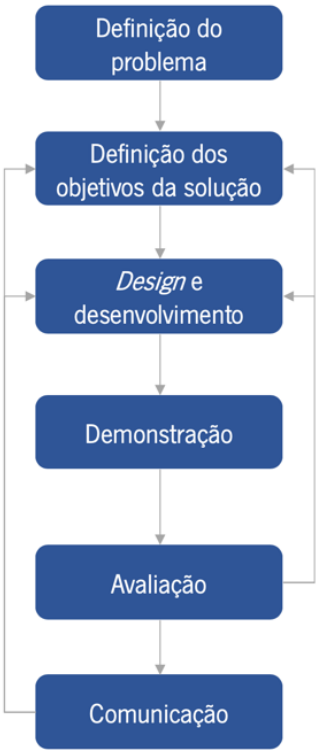
No âmbito da investigação em SI, segundo Hevner et. al. (2004), é possível distinguir-se dois paradigmas de atuação: *Behavioral Science* (cujo objetivo é o desenvolvimento e verificação de teorias que expliquem ou prevejam o comportamento humano ou organizacional) e *Design Science* (cujo objetivo é criar um novo artefacto, que pode ser um algoritmo, constructo, *framework*, instanciação, método ou modelo) (Peffers, et. al, 2012).

A DSR envolve a análise do uso e desempenho dos artefactos, por forma a permitir perceber, explicar e melhorar, entre outros, os SI (Iivari e Venable, 2009). Tem por objetivo o desenvolvimento de novos artefactos, que procuram apresentar uma nova realidade, ao invés de tentar explicar uma realidade já existente (Iivari e Venable, 2009).

Esta dissertação enquadra-se no âmbito de *Design Science*, uma vez que o seu objetivo é o desenvolvimento de um novo método, o AgileMIP. Este novo método tem como objetivo enriquecer a MIP, com uma nova classe de implantação de TI, seguindo os princípios e valores ágeis.

3.3. Processo de investigação

No âmbito desta dissertação foram realizadas as seis atividades do DSR: identificação e motivação do problema, definição dos objetivos da solução, *design* e desenvolvimento, demonstração, avaliação, e, por fim, comunicação (Peffers et al., 2007), conforme representado na Figura 11.



Adaptado de: (Peffers et al., 2007)

Figura 11: *Design Science Research*

Atividade 1: Definição do problema: a compreensão do problema ocorreu através da realização de uma revisão da literatura, sobre o desenvolvimento de SI, diferentes ciclos de vida, modelos de desenvolvimento de software, diferentes perspetivas do *agile*, processo de implantação de TI, e, por fim, sobre *method engineering*.

O processo de implantação é composto por duas dimensões: tecnológica e organizacional (Davenport e Short, 1990; D'Ascenção, 2001; Silveira e Diniz, 2002), influenciada pelas caraterísticas dos projetos e pela sua constante mudança e evolução (Isetta e Sampietro, 2018). Esta dissertação, tem como objetivo o desenvolvimento de um método ágil para a implantação de TI (*cloud* e *on-premise*) da Primavera BSS, motivada pela dificuldade no fecho dos seus projetos, pela necessidade de adaptação às boas práticas realizadas atualmente no mercado, e pela necessidade de responder às exigências dos clientes e parceiros Primavera relativamente à introdução de agilidade no processo de implantação.

Atividade 2: Definição dos objetivos para a solução: depois de obtido o conhecimento necessário sobre o problema e de clarificada a motivação do trabalho, foram detalhados os objetivos, que são: clarificar a aplicabilidade do *agile* no contexto de implantação de TI, comparação de diferentes modelos de desenvolvimento de *software* (com foco nas suas forças e limitações), suportar a implantação de TI da Primavera BSS (*cloud* e *on-premise*), suportar os processos e métodos de trabalho das equipas e

parceiros de implantação Primavera, e a proposta de um novo método de implantação de TI para o caso específico na Primavera BSS, e subsequente generalização.

Após a definição da base do projeto e do artefacto a desenvolver (motivação, objetivo do projeto e objetivos do artefacto), passou-se às atividades de desenho e desenvolvimento (**Atividade 3**), demonstração (**Atividade 4**), e avaliação do artefacto (**Atividade 5**). Aqui foram realizadas quatro iterações, onde, em cada uma, foram aplicadas melhorias consoante o *feedback* obtido na atividade de demonstração e da consequente reflexão realizada na etapa de avaliação.

Atividade 3: *Design* e desenvolvimento: consistiu na identificação das caraterísticas desejadas para o artefacto, e respetiva definição e construção. Aqui, foi elaborada uma pesquisa intensiva sobre o tema *agile* no desenvolvimento de *software*, na gestão de projetos e "agile como *mindset*". Foram também analisados métodos de implantação existentes, tais como a MIP, SAP Activate, Odoo Methodology, e Expertise@Work. Efetuou-se também uma extensiva análise da MIP, e foi observada e experienciada a dinâmica das equipas Primavera no contexto de um projeto de adoção ágil de uma aplicação informática para uso interno. Este projeto de implantação foi acompanhado na etapa de pré-implantação, nomeadamente na etapa de prova de conceito, pressupondo a realização de todo o processo (pré-implantação, implantação e pós-implantação) de acordo com os princípios ágeis.
Para o desenho, foi efetuada uma análise dos elementos mais importantes identificados na literatura, na experiência vivenciada, e nos métodos de trabalho e organização das equipas Primavera, passíveis de acompanhar através da observação e interação com os mesmos.

Atividade 4: Demonstração: consistiu na demonstração do uso do artefacto na resolução do problema, sendo o objetivo verificar se o artefacto criado se adequava ao contexto Primavera. Aqui, procedeu-se à realização de várias apresentações do novo método de implantação, com o objetivo de o apresentar às equipas e obter o seu *feedback* (relativo a preocupações, limitações encontradas, adequação aos métodos de trabalho Primavera, adequação aos projetos, elementos em falta, etc.).

Atividade 5: Avaliação: consistiu na observação do artefacto e do seu comportamento na resolução do problema, comparando os resultados obtidos na atividade de demonstração com os objetivos estipulados.

Atividade 6: Comunicação: a comunicação do problema, da sua importância e do artefacto desenvolvido foi realizada através da presente dissertação, de uma apresentação pública, e da publicação de artigos de natureza científica.

3.4. Estratégia de pesquisa

Nesta secção é descrita a estratégia de pesquisa utilizada para seleção dos artigos, nomeadamente relativamente aos motores de pesquisa utilizados, expressões de pesquisa, número de resultados obtidos e critérios de seleção.

Com vista a responder às questões de investigação, entre outubro de 2019 a maio de 2020 foram efetuadas diversas pesquisas, utilizando os seguintes termos:

1. "information systems", "information systems development";
2. "waterfall model", "spiral model", "incremental model", "unified process", "agile", "Scrum";
3. "method", "process", "model", "methodology", "life cycle", "method engineering";
4. "implementation", "implantation", "adoption";
5. "principles", "values", "guidelines";
6. "advantages", "strengths", "benefits", "limitations", "weaknesses", "constraints", "disadvantages";
7. "empiric", "survey", "case study", "experiment", "field experiment", "interview";
8. "success management", "change management".

Os motores de pesquisa selecionados foram a Web of Science, Scopus, Science Direct, e o Google Scholar, tendo sido utilizados operadores lógicos para a combinação das expressões ("AND", "OR").

As pesquisas realizadas retornaram um elevado número de resultados nos motores utilizados, que contabilizam um total de 305867 resultados. Dado o elevado número de resultados obtidos, verificou-se ser inviável analisar todos os resultados, principalmente porque muitos se revelaram irrelevantes, isto é, vários artigos não se encontravam na área de foco deste trabalho. Tipicamente, tal verificou-se entre a 4.ª e a 6.ª página de resultados de cada motor de pesquisa.

Assim, decidiu-se prosseguir focando nos resultados até à 6.ª página dos resultados de cada motor de pesquisa. A seleção dos artigos ocorreu em quatro etapas:

1. Seleção preliminar: consistiu na análise do título dos artigos, onde deviam constar os termos referidos anteriormente (N=168).
2. Segunda seleção dos artigos: aqui foram analisados o resumo, palavras-chave e a introdução, onde tipicamente se encontra um maior detalhe sobre o artigo. Aqui foram excluídos 73 artigos (N=95).
3. Seleção final: consistiu na leitura integral dos artigos, onde foram excluídos 18 artigos (N=77).
4. Análise das referências bibliográficas: depois de obtida a lista final dos artigos, foram analisadas as referências bibliográficas de cada artigo, onde foram recolhidos outros artigos com relevância.

Depois, foi também realizada uma análise de conteúdo para identificação das forças e limitações dos modelos de desenvolvimento de *software*. O processo foi o seguinte:

1. Leitura integral dos artigos;
2. Identificação das forças e limitações encontradas pelos autores;
3. Compilação das forças e limitações num documento (presente nos Apêndices A.1 ao A.6);
4. Padronização das forças e limitações encontradas;
5. Realização de uma análise comparativa (Apêndices A.7 ao A.9.).

Para além das etapas enumeradas acima, foram também realizadas pesquisas *ad-hoc*, cujo objetivo foi procurar métodos de implantação de TI em utilização nas empresas. Assim, foram analisados os *sites* de algumas das principais empresas internacionais, como a SAP, Oracle, Deloitte, Accenture e a Microsoft. Foram também realizadas pesquisas *ad-hoc*, em português e inglês, com os termos: "método", "processo", "metodologia", "ciclo de vida", "modelo", "ERP", "implementação", "implantação", "ERP", "adoção", "agile", "consultoria", "TI", entre outros.

4 AgileMIP

Nesta secção são apresentados e descritos a Metodologia de Implementação Primavera (MIP) e o novo método criado, o AgileMIP. São também apresentadas as caraterísticas do agile na implantação de TI, e é demonstrada a aplicação prática do AgileMIP no contexto de um projeto Primavera e, consequentemente, a sua avaliação.

4.1. Metodologia de Implementação Primavera (MIP)

Nesta secção é descrita a Metodologia de Implementação Primavera (MIP). É também realizada uma breve apresentação da Primavera BSS e dos seus produtos.

4.1.1. Empresa: Primavera BSS

A Primavera BSS é uma *software house* que define como sua missão "simplificar a vida das organizações" (Primavera BSS, 2020). Dedica-se, por isso, ao planeamento, desenvolvimento e comercialização de aplicações informáticas de gestão, adequadas a qualquer tipo de negócio e dimensão. Para além dessas atividades, fornece aos seus clientes o suporte necessário para que a implantação e a exploração de TI aconteçam com sucesso.

A Primavera BSS tem presença internacional, em Espanha (Madrid), Angola (Luanda), Moçambique (Maputo), e Cabo Verde (Cidade da praia). A nível nacional, está localizada em Braga e Lisboa (Primavera BSS, 2020). Globalmente, conta com cerca de 322 colaboradores, 500 parceiros (revendedores dos produtos Primavera), e mais de 1500 técnicos certificados (Primavera BSS, 2020).

Nas suas áreas de atuação, destacam-se a administração pública, o setor da construção, contabilidade, indústria, restauração e o retalho. Integra, para isto, todas as áreas de gestão, sendo elas a Gestão de Recursos Humanos, Gestão Contabilística e Financeira, Gestão Comercial e de Marketing, *Reporting* Legal e Fiscal, *Reporting* de Gestão, Gestão de Ativos, Gestão de Projetos, Gestão de Materiais e, por fim, Informação Analítica de Apoio à Decisão (Primavera BSS, 2020).

Para a implantação dos seus produtos, utiliza uma metodologia, a MIP, desenvolvida pela própria Primavera, que se encontra descrita na secção 4.1.3.

4.1.2. Aplicações Primavera

A Primavera BSS possui um vasto conjunto de aplicações informáticas que procuram responder às necessidades dos seus clientes. A sua aplicação mais conhecida é o ERP Primavera. Existem, no entanto, muitas outras, dos quais são exemplos o *Eye Peak*, o *Omnia*, *Business Analytics, Personal Data Manager, Primavera Accounting Automation, Primavera Fiscal Reporting,* e o *Office Extentions* (Primavera BSS, 2020). Existe, assim, uma oferta bastante diversificada, para todas as dimensões de uma empresa, tanto do setor público como do privado.

No âmbito deste trabalho importa descrever as aplicações que se relacionam com a MIP, que são todas as aplicações que dão resposta a problemas de empresas de média dimensão, que são (Primavera BSS, s.d):

Eye Peak: solução orientada à gestão multi-armazém, focada na gestão de artigos, rastreabilidade, gestão de clientes e fornecedores, gestão de encomendas e de rotas, receções, expedições, devoluções e outras operações necessárias. Para além destas funcionalidades, acompanha também todas as atividades de armazém, desde a arrumação à entrega.

ERP Primavera: sistema integrado de gestão composto por diversos módulos, integrados nas áreas de compras, contactos e oportunidades, vendas, instrumentos de gestão, recursos humanos, projetos e serviços, inventário, equipamentos e ativos, e contabilidade e tesouraria.

Business Analytics: é uma plataforma analítica, onde se disponibiliza um conjunto de análises e indicadores organizados em *dashboards*. O objetivo é ter "a informação certa, no momento certo para a pessoa certa", otimizando a tomada de decisão. A sua implantação ocorre em cinco etapas: planeamento, análise, realização (podem ser realizadas várias iterações), preparação final, e arranque e acompanhamento.

Primavera *Manufacturing*: aplicação destinada às necessidades do setor industrial, nomeadamente de planeamento e controlo dos custos e tempos de fabrico. A implantação desta aplicação ocorre de acordo com as cinco etapas da MIP, sem existirem iterações sobre as suas etapas (planeamento, análise, realização, preparação final, e arranque e acompanhamento).

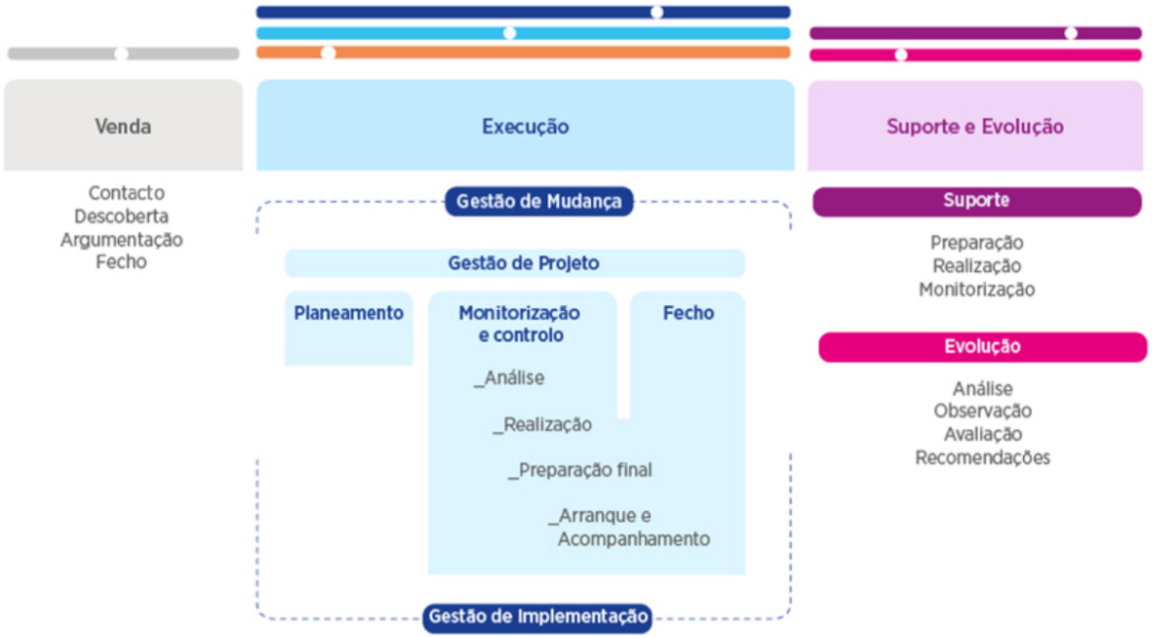
Omnia: plataforma *cloud* (*Deploy as a Service*) para modelação de processos. Permite desenvolver, disponibilizar e gerir aplicações na área dos *Management Information Systems*. A sua implantação segue a MIP, e na terceira etapa (realização) podem ocorrer diversas iterações.

Mostra-se também importante referir o ***Rose***, uma vez que representa um dos motivos de realização desta dissertação. O *Rose* é um novo ERP, ainda em desenvolvimento pela Primavera BSS, que é completamente nativo em *Cloud*.

4.1.3. MIP

A MIP, representada nas Figura 12 e 13, é uma metodologia desenvolvida pela Primavera *BSS,* com o objetivo de uniformizar os processos de implantação de TI, gerir o âmbito dos seus projetos, controlar e diminuir os riscos, garantir sucesso dos projetos, aumentar o foco nos objetivos de negócio e no cliente, aumentar a colaboração, entre muitos outros (Primavera BSS, 2017). Para isso, tem em conta um conjunto de boas práticas estabelecidas para a implantação das aplicações informáticas empresariais,

sendo estas inspiradas pelas técnicas e práticas aceites internacionalmente na implantação e integração de *software* de gestão (Primavera BSS, 2017).

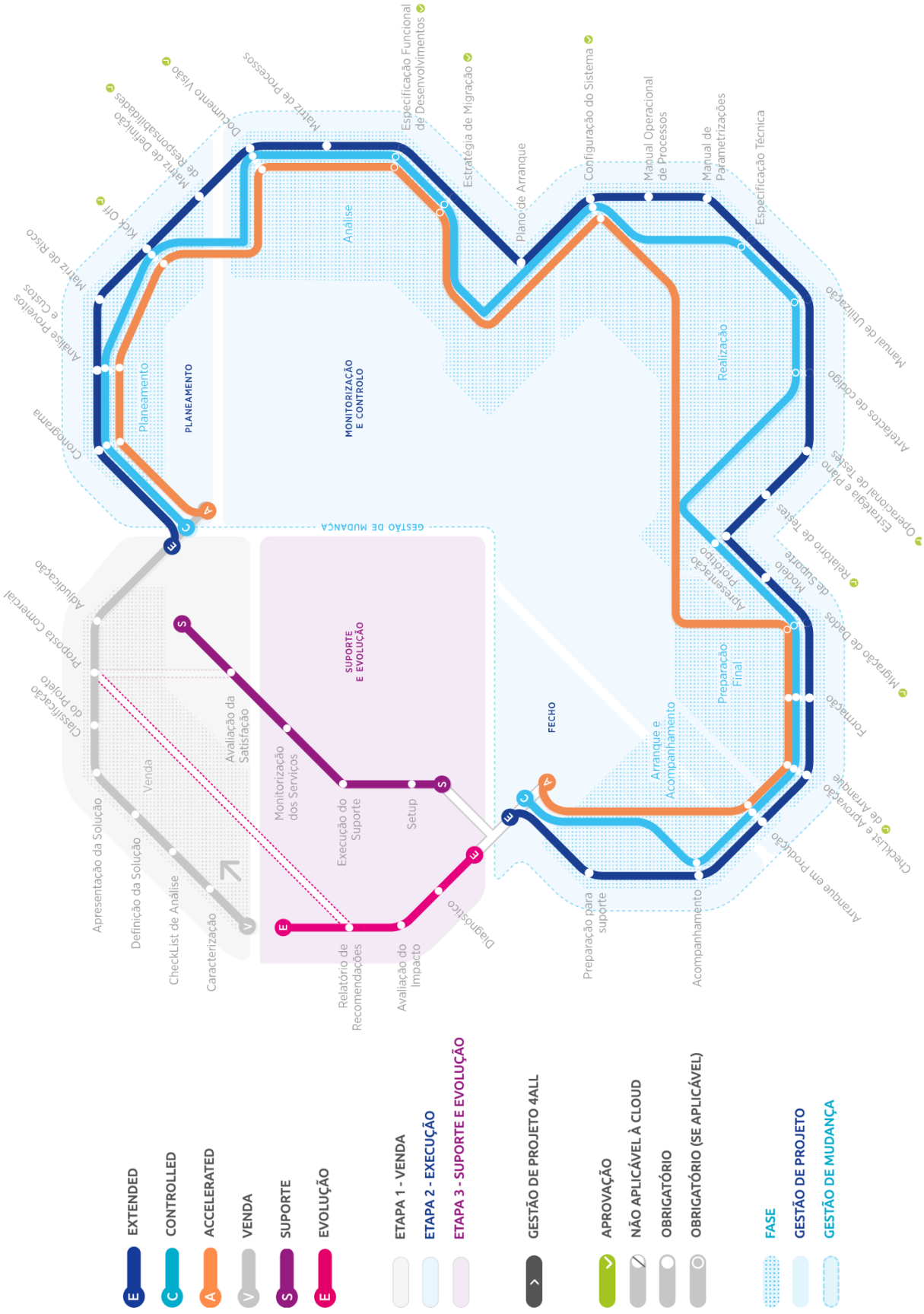


Fonte: (Primavera BSS, 2017)

Figura 12: Metodologia de Implementação Primavera (MIP)

Na MIP são distinguidos três tipos de projetos, que se traduzem em três métodos de implantação (*controlled*, *extended* e *accelerated*), que são atribuídos de acordo com quatro elementos: o cliente (análise do contexto do cliente), o projeto (número de utilizadores, dias de trabalho, dimensão da equipa, etc.), os requisitos (validação da complexidade dos requisitos e dos detalhes da contratualização) e, por fim, os processos (identificação da aplicação a implantar, quais os processos de negócio, integrações e customizações necessárias) (Primavera BSS, 2017).

Assim, a classe ***controlled*** é utilizada quando existe a necessidade de integração entre uma ou mais aplicações, cujas caraterísticas requerem um maior acompanhamento e controlo do processo. Por sua vez, a classe ***extended***, é utilizada quando existe a necessidade de integração com outras aplicações. Habitualmente, enquadram-se nesta categoria projetos onde existe a necessidade de se adequar o produto aos objetivos do cliente e que necessitam de um elevado número de customizações. Finalmente, a classe ***accelerated*** é utilizada aquando a implantação de aplicações *standard*. Estes projetos têm, habitualmente, um âmbito bem delimitado e prazos mais curtos.



Fonte: (Primavera BSS, 2017)

Figura 13: MIP

Conforme representado na Figura 12, a MIP é composta por três fases (Primavera BSS, 2017):

Venda: é a primeira etapa do projeto de implantação e consiste na análise da oportunidade de venda e, consequentemente, na oficialização de um projeto. É composta por quatro etapas: contacto (serve para caraterização da oportunidade), descoberta (identificação dos requisitos do cliente e apresentação e demonstração da solução mais adequada), argumentação (classificação do projeto, requisitos de subcontratação e elaboração, envio e apresentação de uma proposta comercial), e fecho (oficialização da compra).

Execução: consiste na etapa onde ocorre a implantação das TI, através de um processo sequencial, tipicamente em *waterfall*. Aqui, distinguem-se três áreas de trabalho (Primavera BSS, 2017):

Gestão da implementação: tem como objetivo implantar a TI em ambiente de produção e a adequação dos processos de negócio à nova realidade da organização; Gestão do projeto: tem como objetivo gerir a execução do projeto, levando-o ao sucesso, através da previsão de problemas e limitações. Compõe-se por três fases: planeamento (enquadramento do projeto, dos seus objetivos e da sua motivação), monitorização (controlo da execução e progresso do projeto; é composto pelas etapas de análise, realização, preparação final, e arranque e acompanhamento), e fecho (transferência da administração das aplicações informáticas para o cliente). Gestão da mudança: tem como objetivo apoiar os clientes e os seus colaboradores na concretização do sucesso do projeto, através da compreensão das mudanças realizadas nos processos de trabalho.

Suporte e evolução: contempla as atividades necessárias para dar continuidade ao funcionamento das TI implantadas. Inicia-se na fase de preparação final, e é fornecido por uma equipa de suporte, diferente da equipa de execução do projeto.

4.2. *Agile* na implantação de TI

Na secção 2.5. foi analisado o *agile*, focando-o no desenvolvimento de *software*, na gestão de projetos, e enquanto *mindset*. No entanto, embora seja reconhecido na literatura que o *agile* possa ser aplicado a outras áreas do conhecimento (Bishop et al., 2018), não foram encontrados estudos que abordem as caraterísticas do *agile* no contexto de implantação de TI.

Apesar do exposto anteriormente, segundo a Deloitte (2020), as organizações estão cada vez mais a analisar os conceitos associados ao *agile*, para além da sua aplicabilidade no desenvolvimento de *software*. Todavia, de acordo com a análise realizada, existem ainda poucos métodos de implantação de TI que traduzam um ciclo de vida ágil, correspondendo a 16% dos métodos analisados.

Assim, conclui-se, tendo em conta a análise realizada no âmbito desta dissertação, que os métodos de implantação de TI com um ciclo de vida em *waterfall* continuam a predominar, correspondendo a 42% dos métodos analisados. Os métodos de implantação híbridos, por sua vez, possuem também uma

grande adesão por parte das organizações, correspondendo a 26% dos métodos analisados. No entanto, de um modo geral, esses métodos são proprietários havendo relutância por parte das empresas que os criam em abri-los ao domínio público.

Tendo isto em conta, através da análise dos métodos ágeis de implantação de TI encontrados, descritos na secção 2.6.1 desta dissertação, foi possível identificar as características do *agile* no contexto de implantação de TI, respondendo à Questão de Investigação 1: Quais as caraterísticas do mindset ágil no contexto de implantação de TI?

Na Tabela 15 encontram-se as caraterísticas ágeis encontradas em todos os métodos ágeis de implantação identificados na literatura:

Tabela 15: Análise dos métodos ágeis de implantação

#	Caraterísticas dos métodos ágeis de implantação	
1	*Expertise@Work*	Abordagem em *sprints* (4-6 semanas) Conjunto de etapas que são repetidas em cada *sprint* De cada *sprint*, resulta uma parte funcional da aplicação informática Priorização do trabalho a realizar Responsabilidade Consideração de mudança de requisitos e prioridades Abordagem orientada à procura (construir o que é preciso e eliminar tudo o que não adicione valor) Descoberta dos requisitos através de um processo de *design* colaborativo num ambiente interativo Foco nas funcionalidades obrigatórias A mudança pode ser facilmente realizada Garantia do *feedback* do utilizador em todas as tarefas do projeto Envolvimento do cliente
2	*SAP Activate*	Iterações com um conjunto de etapas Abordagem em *sprints* Priorização (obrigatório, deve ter, podia ter e desejo) Adaptação ao cliente Orientação para o cliente Envolvimento do cliente Entrega incremental Validações frequentes do trabalho
3	*Odoo Implementation Methodology*	Ciclo iterativos e incrementais Transparência sobre projeto Foco e adaptação ao cliente, ao seu negócio, necessidade e especificações Envolvimento frequente do cliente Maior colaboração do cliente *Feedback* constante do cliente Comunicação frequente entre todos os elementos do projeto Avaliação e resposta à mudança

#	Caraterísticas dos métodos ágeis de implantação	
		No final de cada iteração existe um conjunto de elementos prontos para utilização em produção

Tendo em conta a Tabela 15, é possível identificar um conjunto de caraterísticas comuns entre os diferentes métodos, que caracterizam o *agile* no contexto de implantação, conforme representado na Tabela 16.

Tabela 16: Caraterísticas dos métodos ágeis de implantação

Características	Métodos	Total
Abordagem em *sprints*	1, 2, 3	3
Entrega incremental	1, 2, 3	3
Adaptação ao cliente, ao seu negócio, necessidades e especificações	1,2, 3	3
Envolvimento do cliente	1,2,3	3
Validação frequente do trabalho realizado	1, 2, 3	3
Cada *sprint* repete um conjunto de etapas	1, 2	2
Priorização do trabalho	1, 2	2
Respeita e aceita a mudança	1, 3	2
O levantamento de requisitos é feito em colaboração	1	1
Orientado ao cliente	2	1
Transparência sobre o projeto	3	1

Assim, pode afirmar-se que as principais caraterísticas do *agile* identificadas no contexto de implantação de TI são as seguintes:

1. Abordagem em *sprints*: realização de iterações que representam um produto, conjunto de produtos ou uma parte do produto.
2. Entrega incremental: no final de cada *sprint* há um incremento, pronto para utilização pelo cliente.
3. Envolvimento do cliente: o cliente é continuamente envolvido no processo de implantação e fornece *feedback* frequentemente sobre o trabalho.
4. Validação frequente do trabalho realizado: antes de o incremento ser entregue existe uma demonstração da *sprint* e uma validação desse trabalho pelo cliente.

Existem também outras caraterísticas que se realçam na Tabela 16, tais como a priorização do trabalho, e o respeito e aceitação da mudança.

Posto isto, pode afirmar-se que existem várias caraterísticas comuns ao *agile* no desenvolvimento de *software*, que se relacionam, maioritariamente, com a forma de organização do trabalho e de envolver o cliente. Isto vem corroborar a perspetiva do *agile* enquanto *mindset*, que congrega o foco na colaboração, entrega, reflexão, e na melhoria (Cockburn, 2016).

4.3. AgileMIP

Ao longo deste trabalho foram vários os métodos de implantação de TI estudados, os quais possuem princípios e valores que os caraterizam. O âmbito deste trabalho enquadra-se, no entanto, no *agile*, um *mindset* onde são defendidos valores, como o trabalho em equipa, a entrega e melhoria contínua, a cooperação e envolvimento constante do cliente, a flexibilidade e a contínua priorização (Rubin, 2012; PMI, 2017; Agile Alliance e PMI, 2017; Stellman e Greene, 2014).

O método de implantação de TI criado, designado de AgileMIP (Figura 14), tem como objetivo trazer os princípios e valores do *mindset agile* para a implantação dos produtos Primavera (*on-premise* e *cloud)*. Este novo método, foi concebido tendo por base os diversos métodos de implantação estudados, as boas práticas utilizadas no mercado (tais como o PMBOK (PMI, 2017), o Prince 2 *Agile* (Cooke, 2016; Carvalho, 2018) e o PM² (EU, 2016), e a dinâmica de trabalho das equipas de consultoria Primavera.

Antes da descrição e apresentação do AgileMIP, é pertinente esclarecer alguns conceitos incorporados neste método. Vários são os elementos provenientes de diferentes fontes (referenciais, métodos, *frameworks*, relatórios, etc.) que inspiraram e estão presentes no AgileMIP, tais como:

- MIP (Primavera BSS, 2017): a MIP representa a atual forma de trabalho das equipas de consultoria Primavera e dos seus parceiros. Sendo um dos objetivos do AgileMIP respeitar a forma de trabalho e organização destas equipas, mostrou-se importante analisar e utilizar alguns dos seus elementos, que são: gestão da mudança, etapa de venda, reunião de *kick-off*, *deliverables* (como, por exemplo, o documento de caraterização global do negócio, pedido de mudança e documento de *kick-off*), e a etapa de fecho;
- *Scrum* (Beck et al., 2001): foram integradas as diversas cerimónias identificadas no *Scrum*, o conceito de *sprint,* e os seus três princípios: transparência, inspeção e adaptação;
- PMBOK (PMI, 2017): o grupo de processos de iniciação, descritos no apêndice X3.3 do PMBOK, consistem em processos cujo objetivo é definir um novo projeto ou fase (ou iteração), através da obtenção de uma autorização para iniciação dessa fase ou projeto. No caso do AgileMIP, este conceito está desdobrado por duas etapas: na preparação (obtenção de um conhecimento base comum sobre o projeto e de como este vai decorrer), e, em cada iteração, através da etapa de definição (visão para cada iteração);

- PM² (EU, 2016): neste referencial, conforme descrito no Apêndice D.1: PM² *and Agile Management*, as iterações ocorrem apenas na execução. Esta lógica foi também adotada no AgileMIP;
- Prince2 Agile (Cooke, 2016): classificação do tempo, custo, âmbito, qualidade, benefícios, e riscos, em três caraterísticas (não flexível, flexível ou podem ser flexíveis). Esta classificação é adotada no AgileMIP – o tempo e o custo são fixos e não se alteram ao longo do projeto, o âmbito e a qualidade são flexíveis e existe uma grande probabilidade de variarem ao longo do projeto, e os benefícios e os riscos podem ser flexíveis e ajustados à medida que o projeto evolui.
- SAP Activate Methodology (SAP, 2018): este método de implantação do ERP SAP encontra-se bem estruturado, no que relativo às atividades, ao processo e às nomenclaturas escolhidas, tendo sido, por isso, analisado para o desenvolvimento do AgileMIP;
- D'Ascenção (2001): definição da atividade e processo de redesenho dos processos de negócio.

O objetivo do AgileMIP não é substituir a MIP, apresentada na secção 4.3 desta dissertação, mas sim complementá-la com uma nova classe de projeto, onde também se encontram as classes *accelerated, controlled,* e a *extended*. Tal como as restantes classes, também a classe *agile* requer que os projetos possuam determinadas caraterísticas para que seja possível alcançar o máximo sucesso, que se encontram em concordância com a análise realizada nos Apêndices A.7 ao A.9, que são:

- O projeto deve ser de pequena ou média dimensão;
- Adequado a projetos com algum grau de incerteza e risco face ao que é necessário realizar;
- O projeto não exige um elevado grau de definição *a priori*;
- A equipa deve ser auto organizada, multifuncional, e de pequena dimensão (Schwaber e Sutherland, 2017);
- Há disponibilidade do cliente para acompanhar e estar presente ao longo do projeto;
- Existe a possibilidade de realização de reuniões frequentes com o cliente;
- É possível priorizar o trabalho de acordo com a necessidade do cliente;
- Há abertura à mudança;
- Espera-se entrega contínua e frequente desde cedo no projeto;
- Há foco no cliente.

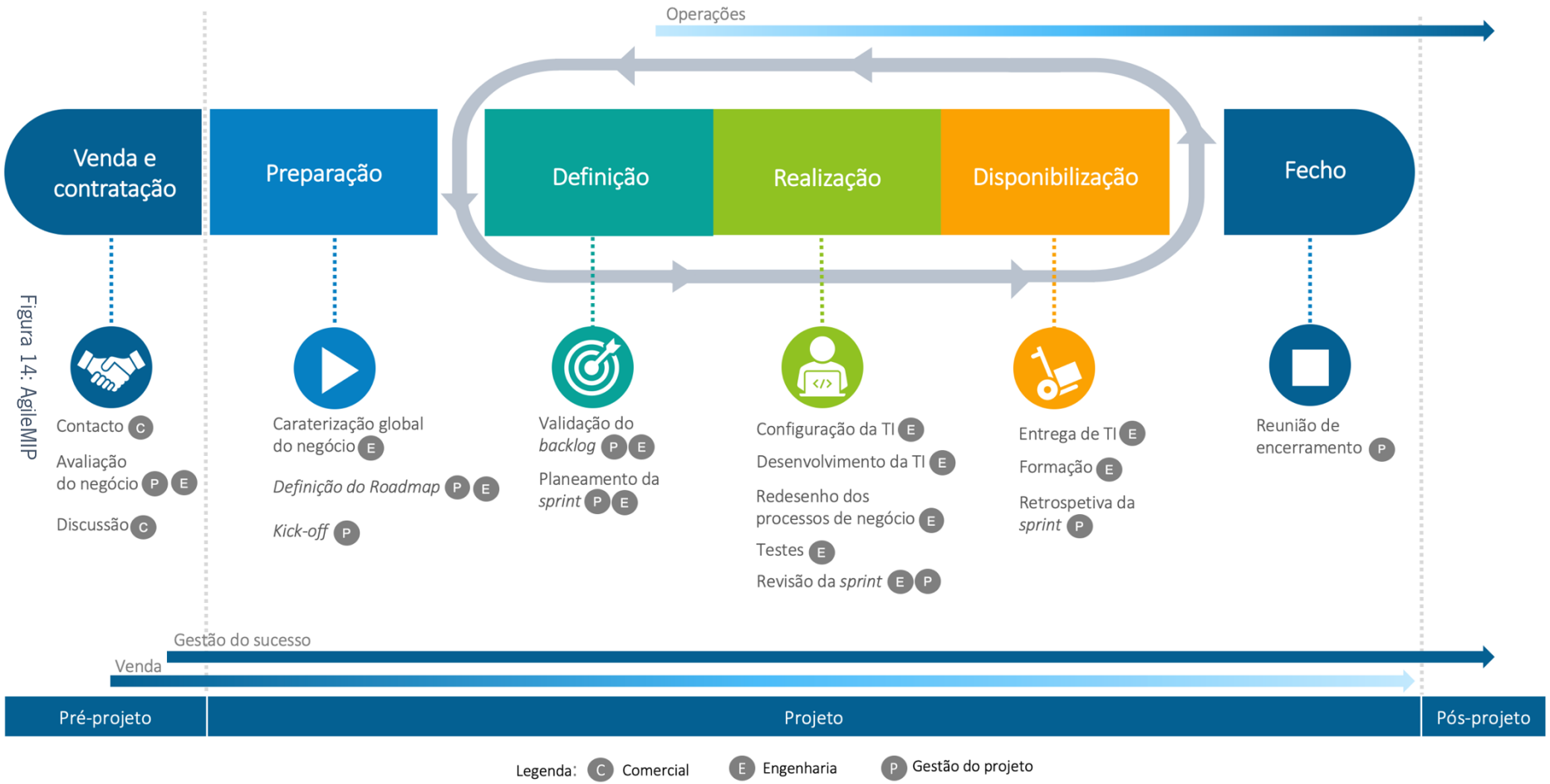


Figura 14: AgileMIP

Dadas estas considerações, o AgileMIP pode ser dividido em três momentos: pré-projeto (momento de venda e contratação), projeto (execução do projeto, composto pela preparação do projeto e pela realização de ciclos iterativos e incrementais) e pós-projeto (momento onde ocorre a finalização do projeto, sendo o seu objetivo apoiar o cliente depois do fecho do projeto). É importante realçar que, no decorrer do projeto, novos projetos podem surgir, que seguirão o fluxo descrito.

4.3.1. Pré-projeto

A fase de pré-projeto é composta por uma etapa, de **venda e contratação**, que consiste no primeiro contacto com o cliente. É, assim, composta por três atividades:

- Contacto comercial: consiste em compreender o cliente, o seu negócio, a sua organização e, principalmente, o seu problema;
- Avaliação do negócio: traduz-se na análise da situação atual do negócio do cliente. Aqui deverá ocorrer o levantamento de *user stories* e consequente priorização, de acordo com a necessidade e desejo do cliente.
- Discussão: consiste em conversas com o cliente sobre a abordagem a seguir no projeto, limitações identificadas, prazos de trabalho, entre outros elementos que se mostrem pertinentes para um determinado projeto.

Relativamente aos *deliverables*, são produzidos os seguintes documentos:

- *Backlog*: documento onde constam o conjunto de *user stories* a trabalhar ao longo do projeto. Nesta etapa, o *backlog* apenas conta com um número de requisitos iniciais. Isto significa que, no decorrer do projeto, poderão ser acrescentados, removidos ou atualizados *user stories*. Espera-se, no entanto, que seja o mais completo que for possível.
- Proposta comercial: documento através do qual é descrito o que se pretende realizar no projeto, tais como atividades, datas, limitações, perspetivas, entre outros elementos que se justifiquem para um projeto em específico.
- Contrato: documento de oficialização do projeto.

4.3.2. Projeto

A fase de projeto é composta por cinco etapas: preparação, definição, realização, disponibilização e fecho.

A etapa de **preparação** consiste numa avaliação mais detalhada do trabalho a realizar com o cliente, depois da oficialização da realização do projeto. Esta etapa é composta por três atividades:

- Caraterização global do negócio: esta atividade é realizada pelo consultor e consiste num período onde um consultor se desloca às instalações do cliente para o conhecer e dar-se a conhecer,

bem como conhecer os processos de negócio, o negócio, métodos de trabalho, e identificar possíveis especificidades. O seu objetivo é proporcionar um maior detalhe sobre o ambiente e o contexto onde a implantação da TI irá decorrer.

- Definição do *roadmap*: esta atividade é realizada pelo gestor de projeto e pelo consultor. Consiste na divisão dos *user stories* em grupos de trabalho, criados consoante a prioridade do cliente no momento da sua elaboração. Esta etapa consiste na definição do *roadmap* do projeto, a partir do qual se pode aceder a uma visão global do projeto, uma das limitações apontadas ao *agile* na literatura, conforme pode ser visualizado na Tabela 12. Este *roadmap* representa, no entanto, apenas uma previsão.
- *Kick-off*: atividade que oficializa o lançamento do projeto, onde são identificados e apresentados a equipa do projeto, as datas acordadas, os custos e todos os restantes elementos definidos associados ao projeto. Consiste numa reunião com a equipa Primavera, o cliente e a sua equipa, onde é apresentado o documento de visão global do projeto.

A etapa de **definição** consiste na identificação e análise no trabalho a realizar durante uma *sprint*. Nesta etapa, procura-se também aprimorar as *user stories* anteriormente identificados, bem como identificar mudanças, atualizações ou novas necessidades. É composta por duas atividades:

- Validação do *backlog*: esta atividade envolve o gestor do projeto, o consultor e o cliente. Consiste na revisão do trabalho a realizar na *sprint,* face ao que foi inicialmente identificado. Aqui, procura-se identificar novas necessidades, requisitos ou alterações por parte do cliente, bem como incluí-lo no processo de planeamento da *sprint*. Assim, o objetivo desta atividade é perceber se existem desvios face às necessidades e aos requisitos inicialmente identificados.
- Planeamento da *sprint*: inspirada no *Scrum*, esta atividade é realizada pelo gestor do projeto e pela equipa do projeto, tendo em consideração o trabalho realizado com o cliente na validação do *backlog*. Consiste no planeamento do trabalho a realizar na *sprint* de acordo com a priorização referida pelo cliente e das dependências funcionais identificadas da aplicação informática. Consiste também em perceber, conjuntamente com a equipa do projeto, quanto tempo será necessário para realizar as tarefas da *sprint*.

A **realização** consiste na realização do trabalho planeado para uma *sprint*, e é composta por quatro atividades:

- Configuração de Tecnologias da Informação (TI): configuração da TI, tendo em conta o objetivo e os requisitos identificados para a *sprint.*

- Desenvolvimento de TI: nesta atividade são realizados os desenvolvimentos à medida (quando necessário), bem como integrações, migrações, e outras atividades necessárias à operacionalização do incremento. Os desenvolvimentos à medida devem ser cuidadosamente descritos e apresentados ao cliente antes de serem colocado em ambiente de produção.
- Redesenho dos processos de negócio: desenho de novos processos de negócio, com base numa análise *AS-IS*, *OUGHT TO BE* e *TO BE* do processo em estudo, que garanta o melhor funcionamento da organização e um melhor *fit* com a nova realidade a implantar.
- Testes de TI: esta atividade é obrigatória e consiste no teste das configurações e/ou desenvolvimentos realizados. Idealmente, esta atividade é realizada não só por quem realizou as configurações e desenvolvimentos, como também por quem for pertinente participar.
- Revisão da *sprint:* inspirada no *Scrum*, esta cerimónia consiste numa reunião entre o gestor do projeto e o cliente, onde este aprova, ou não, o trabalho realizado ao longo de uma *sprint*.

A etapa de **disponibilização** consiste na disponibilização de um incremento totalmente funcional para o cliente em ambiente produtivo, o qual poderá ser utilizado, desde este momento, nas suas atividades diárias. Este incremento, pode ser uma funcionalidade, um documento ou qualquer outro elemento que o cliente considere essencial para o projeto avançar. Assim, o incremento será o resultado de uma *sprint*, e corresponderá sempre a um elemento completamente funcional (TI, processo, etc.), pronto a utilizar em ambiente de produção. Esta etapa é composta por três atividades:

- Entrega de TI: corresponde à entrega de *software*, *hardware* ou infraestrutura e da sua operacionalização no contexto organizacional do cliente, garantindo o correto *fit* entre a tecnologia e a organização.
- Formação: consiste em preparar os colaboradores da equipa do cliente a utilizar e adaptar-se à nova TI e forma de trabalhar introduzida na organização. Esta atividade, e as suas especificações, é organizada conjuntamente pela equipa Primavera e do cliente, por forma a que decorra da melhor forma.
- Retrospetiva da *sprint*: inspirada no *Scrum*, esta cerimónia, que ocorre a nível interno com a equipa Primavera, tem como objetivo identificar quais os obstáculos identificados ao longo da *sprint*, se foram resolvidos, como foram resolvidos, e, finalmente, sugerir e identificar melhorias a aplicar na próxima *sprint*.

As etapas definição, realização e disponibilização são realizadas iterativamente, isto é, uma iteração compõe-se destas três etapas. Cada iteração tem uma duração curta, entre duas a quatro semanas. No contexto do AgileMIP, uma iteração é denominada de *sprint* (designação tipicamente utilizado no *Scrum*),

e nela pode ocorrer a configuração ou desenvolvimento de 1: tarefas, selecionadas e realizadas de acordo com a priorização do cliente.

Durante cada *sprint*, são também realizadas reuniões diárias, também inspiradas no *Scrum*. Nestas cerimónias são reunidos todos os elementos da equipa Primavera envolvida no projeto, onde é realizada uma reflexão sobre o trabalho realizado, as limitações encontradas e o trabalho a realizar até à próxima reunião.

No final de cada *sprint,* por sua vez, ocorre uma demonstração do trabalho realizado ao cliente, presencialmente ou remotamente. Os resultados, que devem ser apresentados a todos os *stakeholders* relevantes para o projeto, são apresentados em dois momentos distintos: internamente – na retrospetiva da *sprint* – e externamente – na revisão da *sprint*.

A etapa de **fecho** corresponde à última etapa do projeto e consiste no seu término, isto é, não serão realizadas mais *sprints*. Aqui, apenas ocorre uma atividade: a reunião de encerramento do projeto.

Nesta reunião estão presentes todos os envolvidos no projeto. O objetivo é realizar uma reflexão sobre todo o projeto, nomeadamente sobre os diversos obstáculos encontrados ao longo das *sprints*, como foram superados e quais os pontos onde se pode melhorar. Deve também ser identificado o que correu melhor e o que poderia ter sido diferente, por forma a aplicar esse conhecimento nos próximos projetos. Esta reunião é também inspirada na cerimónia de retrospetiva, realizada à *sprint*, proveniente do *Scrum*.

Para além destes elementos, deve também ser apresentado o *backlog* final do projeto, onde devem ser identificadas as tarefas que foram realizadas com sucesso, as que foram removidas, e as que não se realizaram, juntamente com as respetivas justificações. Sempre que se mostre pertinente, devem também ser feitas recomendações ao cliente. Toda esta informação deve ser descrita no documento de fecho do projeto.

Ao longo da fase do projeto, são produzidos diversos *deliverables*. A sua identificação foi realizada através da realização de uma análise dos documentos elaborados consistentemente na maior parte dos projetos de implantação da Primavera BSS, que foram: documento de visão global de processos, documento de *kick-off* do projeto e a folha de presenças em formação.

Estes documentos foram contemplados no AgileMIP, embora alguns tenham significados e nomenclaturas diferentes. O documento de visão global de processos é um exemplo. Este documento é oriundo da etapa de análise realizada na fase de execução da MIP, retrata um processo intensivo de caraterização de todos os processos do negócio do cliente, feito *a priori* e descrito através de diagramas em UML.

No AgileMIP existe também uma preocupação em conhecer os processos de negócio do cliente. Inicialmente, na etapa de preparação, procura-se conhecer o negócio do cliente e as suas especificidades, dando-se prioridade aos processos mais críticos do negócio. No entanto, não é procurado caraterizar todo o negócio *a priori*, uma vez que existe um maior espaço para conhecer e interagir com o cliente e o seu negócio ao longo do projeto.

Assim, na etapa de preparação é iniciado o documento de visão global do projeto, procura-se descrever as necessidades específicas do cliente, as suas caraterísticas e processos organizacionais mais críticos, e o que se pretende fazer com o projeto, quando e como (*roadmap*). Este documento, traduz apenas uma previsão, uma vez que existem avaliações diárias do trabalho a realizar e um planeamento do trabalho realizado à *sprint,* contando sempre com a participação e envolvimento do cliente. Comparativamente ao documento de visão global de processos, mostra-se um documento mais sucinto, que vai sendo refinado, completado e atualizado consoante o trabalho do *backlog*.

Para além destes *deliverables*, o AgileMIP considera outros:

- O documento de *kick-off*, elaborado na etapa de preparação pelo consultor Primavera, tem como objetivo apoiar a reunião de *kick-off*. Aqui são colocadas as informações presentes no documento de visão global do projeto, de uma forma mais apelativa e simplificada.
- O pedido de mudança (ou *change request*), tipicamente elaborado na etapa de definição, consiste numa descrição de uma mudança a realizar no projeto e é um comprovativo de que o cliente solicitou e aceitou a mudança e que compreende o seu impacto e implicações no projeto. Este documento não é elaborado para qualquer mudança no projeto, se não para mudanças que representam um grande impacto no projeto, em termos de custo, tempo, âmbito ou outra variável considerada crucial no projeto.
- A declaração de aprovação ou rejeição da *sprint* é elaborada na etapa de realização, nomeadamente na atividade de revisão da *sprint*. Esta declaração pode ser realizada de diferentes formas: pode ser um documento ou uma *checklist* na plataforma de gestão de projeto utilizada pela Primavera, o *Triskell*. O seu objetivo é registar a validação do cliente face ao trabalho realizado na *sprint*. No caso de rejeição, deve ser identificado o motivo e, se adequado, a forma como será resolvida e o seu impacto.
- As especificações de desenvolvimento são elaboradas na etapa de realização, sendo proveniente da atividade de desenvolvimento de TI. Neste documento é descrito o desenvolvimento em termos da sua funcionalidade, aspetos técnicos e contexto.
- O incremento de TI é o resultado do trabalho realizado numa *sprint*, entregue na etapa de disponibilização, na atividade de entrega de TI. Este incremento pode corresponder a *software*,

hardware ou infraestrutura ou a especificações do trabalho a realizar (configurações, desenvolvimentos, alterações dos processos de negócio, entre outros), sem os quais o cliente não avança no projeto.

- A lista de presenças é elaborada na atividade de formação realizada na etapa de disponibilização. Serve como comprovativo de que a formação foi fornecida e sobre quem a frequentou (formador, participantes, empresa cliente, etc.).
- Por fim, o documento de fecho de projeto, elaborado na etapa de fecho, procura identificar o trabalho que inicialmente foi planeado, mas que não foi concretizado no decorrer do projeto, assim como incluir as respetivas justificações e as recomendações face ao identificado.

Estes são os *deliverables* sugeridos no AgileMIP, no entanto estes documentos não são de caráter obrigatório, rígidos ou inflexíveis. Tal como em todo o AgileMIP, é defendida a flexibilidade e adaptabilidade, pelo que devem ser elaborados todos os documentos que se mostrem essenciais ao projeto ou que sejam solicitados pelo cliente.

Destaca-se que este método de implantação de TI se insere num contexto de prestação de um serviço, pelo que se mostra necessária a existência de elementos que comprovem a aceitação, conhecimento e verificação do trabalho e das decisões tomadas num projeto.

4.3.3. Pós-projeto

No final da etapa de fecho, a última etapa do AgileMIP, o projeto termina, mas assegurando a continuidade do trabalho, caso contrário o cliente, apesar de ter experienciado um acompanhamento contínuo ao longo do projeto, sentir-se-ia sozinho aquando o seu término. Assim, o projeto entra em suporte pelo menos durante um período determinado, contratado entre a Primavera BSS (ou o seu parceiro) e a empresa cliente.

Para além dos elementos anteriormente apresentados, existem ainda outros elementos de elevada importância, que são:

- Operações: no âmbito do AgileMIP as operações consistem em duas atividades – suporte e monitorização, que ocorrem continuamente desde a etapa de definição.

  O **suporte** é uma atividade que está presente desde a disponibilização do primeiro incremento ao cliente e visa auxiliar e esclarecer o cliente na utilização da TI, através da resolução de problemas ou anomalias de funcionamento, esclarecimento de dúvidas, explicações, entre outras situações que possam surgir. A atividade de suporte pode, no entanto, levar a que surjam novos projetos, sempre que as situações apresentem uma grande

complexidade associada, tal como um desenvolvimento significativo, estando, portanto, representada através da seta venda, na Figura 14.

Uma outra atividade, com grande importância, é a **monitorização**. No AgileMIP, a monitorização ocorre de maneira relativamente diferente face aos restantes métodos (tipos de projeto) da MIP, uma vez que é realizada de uma forma mais informal. Isto é, como existe uma maior comunicação, proximidade e mais momentos de avaliação e reunião com a equipa e o cliente (propiciados pelas diversas cerimónias existentes no método), há uma maior e melhor noção do projeto, das expectativas, do que cada um está a fazer e de qual o ponto de situação, bem como dos problemas, desafios e mudanças que poderão estar a ser enfrentados e que influenciarão o projeto.

- Gestão do sucesso: procura assegurar o sucesso da gestão do projeto e do sucesso dos *deliverables* do projeto (Varajão, 2016; Varajão, 2018).
- Gestão da mudança: consiste numa abordagem cíclica para a transição da organização para um novo estado, cujo objetivo é auxiliar a organização cliente a alinhar pessoas, processos, estruturas, culturas e, eventualmente, a sua estratégia (PMI, 2013).

4.4. Aplicação e avaliação do AgileMIP

No âmbito deste trabalho estava prevista a aplicação do AgileMIP no contexto de um projeto real Primavera. No entanto, dada a conjuntura vivenciada no momento de elaboração deste trabalho de dissertação, de pandemia, tal não aconteceu na sua totalidade.

Tendo estes fatores em conta, a avaliação do AgileMIP assumiu outra dimensão:

- Elaboração de uma previsão do decorrer do projeto,
- Apresentação e discussão do método às equipas Primavera,
- Comparação com outros projetos de *Eye Peak*.

O projeto selecionado para a aplicação do AgileMIP, foi um projeto de implantação da aplicação informática *Eye Peak*, que, conforme já apresentado no Capítulo 4 desta dissertação, é uma aplicação informática orientada à gestão multi-armazém.

O primeiro passo, antes da aplicação do AgileMIP, é conhecer a aplicação informática em termos das suas dependências funcionais, através da elaboração de uma rede de dependências. A sua elaboração advém da preocupação associada à implantação parcelar da TI, relativa, por exemplo, à necessidade de realização de várias integrações. Assim, o seu objetivo é identificar as dependências funcionais inerentes a uma TI, e, dessa forma, as diversas possibilidades de a implantar.

Este trabalho permitiu perceber quais os módulos e funcionalidades que compõe o *Eye Peak,* e, dessa forma, criar uma priorização inicial baseada na estrutura funcional da aplicação, conforme representado na Figura 15.

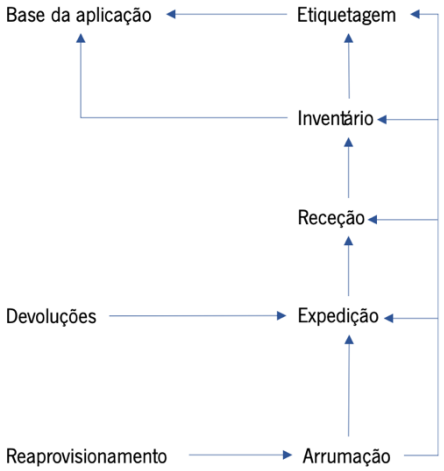


Figura 15: Rede de dependências do *Eye Peak*

A rede de dependências foi elaborada conjuntamente com profissionais Primavera com experiência na implantação deste produto. Foi, depois de elaborada, confirmada ao longo das diversas apresentações com diferentes profissionais intervenientes, não tendo sido realizadas alterações à rede inicialmente elaborada.

Depois de obtidas as dependências funcionais do *Eye Peak*, iniciou-se o planeamento do projeto. O trabalho elaborado corresponde a uma previsão de como decorreria o projeto com o AgileMIP. Realça-se que, no contexto real de um projeto, o planeamento detalhado do projeto não é realizado *a priori*. Existe uma visão global do projeto na etapa de preparação, que colmata uma das limitações apontadas ao *agile* na Tabela 12 (a perda da visão global do projeto), no entanto o planeamento é realizado em cada *sprint*, conforme a priorização do cliente e as dependências da aplicação.

Depois de elaborada a primeira versão do planeamento, procedeu-se à apresentação do AgileMIP a diferentes elementos das equipas Primavera, tais como diretores, gestores de projeto e consultores, tendo sido realizados cinco eventos:

1. Reunião com a Diretora da área da Consultoria (CSU): A primeira reunião para apresentação do AgileMIP foi com a diretora da CSU da Primavera BSS. Aqui, foi sentida uma enorme recetividade ao novo método, bem como aos princípios e valores do *mindset* ágil. A diretora afirmou que gostou do método, que percebeu bem a sua mensagem e a lógica associada.
2. Reunião com um *Delivery Manager*, um *Lead Consultant* e com a Diretora da CSU: Nesta reunião, com um *Delivery Manager* e um *Lead Consultant*, foi sentido um pouco de receio e preocupação relativamente ao *agile* e à sua aplicabilidade no contexto de implantação dos produtos Primavera.

Os dois elementos desta reunião, contrariamente à Diretora da CSU, mostraram-se relutantes quanto ao novo método, defendo que deveria continuar a existir uma entrega única (no final do projeto), bem como uma análise extensiva de todo o projeto *a priori.* Mostraram também preocupações com o custo do projeto e o âmbito.

O *Lead Consultant* sugeriu a criação de um MVP (*Minimum Viable Product*) e a consideração de margens (tanto para o caso de mudança de âmbito, como para o desenvolvimento).

3. Reunião com dois gestores de projetos: Nesta reunião um dos gestores de projetos eram especializados: um com mais experiência em *Eye Peak* (Sujeito A) e outro com mais experiência na implantação do ERP (Sujeito B). O *feedback* obtido nesta reunião, contrariamente à anterior, foi significativamente mais positivo. Ambos os gestores de projetos vêm a aplicabilidade do novo método. Em especial, o Sujeito B ficou bastante entusiasmado, uma vez que já tinha lido e assistido a conteúdos sobre o tema, e, inclusive, já tinha refletido sobre a aplicabilidade do *agile* na implantação do ERP. Mostrou-se, também, com vontade de participar no projeto, especialmente na elaboração da rede de dependências do ERP. Apesar da abertura ao novo método, e, de uma forma geral, concordarem com o foi realizado, apresentaram também algumas preocupações:
   - O Sujeito A apenas vê a aplicabilidade do *agile* em projetos pequenos;
   - Tanto o Sujeito A, como o Sujeito B, mostraram preocupações com o esforço (qual a diferença entre os métodos clássicos, que já conhecem, face ao AgileMIP).
4. Reuniões com o *Agile Coach*: O *Agile Coach* da Primavera BSS foi um elemento que foi acompanhando o desenvolvimento do AgileMIP quase desde o seu início, através do esclarecimento de conceitos, validação de ideias, sugestões, entre outros. Foram, por isso, realizadas várias reuniões e conversas. As duas últimas reuniões foram particularmente importantes, pois o método já tinha incorporado algumas das sugestões obtidas nas reuniões, explicitadas anteriormente, e, por isso, o AgileMIP encontrava-se já próximo da sua versão final.

   Nesta reunião, similarmente às restantes, foi realizada uma apresentação do AgileMIP, e, depois, apresentado o *roadmap* do projeto. O *Agile Coach* sugeriu a melhoria da apresentação do método, sugerindo a elaboração de uma representação similar à do *Scrum*, afirmando que simplificaria a interpretação do método e o tornaria mais percetível. Sugeriu, também, a separação de *deliverables* das restantes atividades, uma vez que considera esta arrumação confusa.
5. Envio do *roadmap* a outros colaboradores: Todas as outras reuniões foram realizadas, na maior parte dos casos, presencialmente. No entanto, dada a situação de pandemia, a Primavera BSS colocou os seus colaboradores em teletrabalho, sendo, por isso, impossível a continuidade dessas reuniões. Outro fator, foi a indisponibilidade dos colaboradores para realizarem reuniões nesta

altura, uma vez que se encontravam bastante ocupados. Assim sendo, optou-se pelo envio da apresentação do AgileMIP e do *roadmap* do projeto de *Eye Peak* para alguns colaboradores:

- *Business Developer:* colocou diversas questões relativamente ao método. Questiona a reduzida quantidade de documentação produzida, a capacidade de resposta à mudança de requisitos e de âmbito, e ainda mostra preocupação relativamente às dependências funcionais da aplicação. O AgileMIP responde a todas as questões colocadas.

  Considera também que o tempo atribuído a cada *sprint* é curto, afirmando que "nunca é possível realizar tarefas corridas nos clientes por falta de disponibilidade". Defende também que o *Go-live* deve ser único, e faz a assunção de que "quando a entrega é realizada em *sprints* o tempo é maior". Outro ponto focado no comentário é a necessidade de visualizar os processos como um todo, assumindo que a análise é realizada à *sprint*. Por fim, afirma que consegue visualizar melhor a aplicabilidade das *sprints* no ERP, sugerindo a divisão por módulos. Termina o seu comentário com a afirmação de que necessita de formação em *Agile*.

  Nota do investigador: as diversas questões relativamente ao método podem justificar-se pelo facto de este profissional não conhecer o AgileMIP. Relativamente à necessidade de visualização dos processos como um todo, no AgileMIP existe uma visualização global do projeto, que não necessita de ser extensiva e meticulosamente detalhada, uma vez que existe um maior espaço para conhecer o cliente, o seu negócio e os seus processos, através do seu maior envolvimento e da contínua comunicação. Este participante afirma que visualiza melhor a aplicabilidade da realização de sprint no contexto do ERP Primavera, no entanto, o caso do ERP é bastante mais complexo do que o *Eye Peak*, não sendo trivial a aplicabilidade do conceito, isto é, por ser uma aplicação maior e mais complexa, requer um trabalho mais afincado e com maior detalhe.
- *Project Manager*: defende a existência de apenas uma entrega, algo que acontece tipicamente nas abordagens em *waterfall*. Sugere a existência de um ambiente de testes, onde a aplicação seria validada recorrentemente. Esta solução, no entanto, não vai de acordo com a lógica ágil, que defende a disponibilização de um produto totalmente funcional para o cliente utilizar.
- *Senior Lead Consultant*: este consultor sénior considera que o produto *Eye Peak* deveria estar mais maduro e não considera o tempo atribuído descabido para implantações *standard*, que é o caso do *roadmap* elaborado. Considera que a formação deveria ser um pouco mais alargada, e afirma que, habitualmente, "valida o trabalho realizado com o cliente durante a formação". Refere também que é perdido muito tempo em anomalias de comportamento da aplicação e na deteção e correção de *bugs*. Sugere a previsão de um tempo para a realização destas alterações e, também, que projetos mais complexos devem ter uma avaliação própria.

Nota do investigador: Estas caraterísticas foram incorporadas no AgileMIP, inspirada no Prince2 Agile, onde é aconselhada a consideração de tolerâncias para as atividades de um projeto, garantindo uma margem para, por exemplo, atrasos ou outros imprevistos.

- *Delivery Manager*: este *Delivery Manager* é o mesmo da reunião 2. Aqui, sugeriu detalhar no *roadmap* as tarefas que seriam realizadas *on* e *off-site*.
- *Consultant A*: considera ótimo que exista uma abordagem de implantação *agile* e concorda a 100%, inclusive afirma que já tinha feito essa sugestão no passado. No entanto, acha que 30 dias é insuficiente para a implantação de um projeto *Eye Peak*. Afirma, no entanto, que o facto de pensar dessa forma poderá estar associado ao facto de ainda "estar muito habituado ao *waterfall,* e não tanto ao *agile*".
- *Consultant B*: considera que o tempo total do projeto para implantação do *Eye Peak* é insuficiente. Realça que muitas vezes surgem entraves que, com os atuais métodos, acabam por atrasar o projeto. Considera, similarmente a outras opiniões, que dois dias de formação não é suficiente para projetos de maior dimensão, apesar de afirmar que para projetos mais pequenos, poderá ser suficiente. Concorda com a divisão em *sprints* e faz algumas sugestões:
    1. Contemplar melhorias e correções do produto;
    2. Realização de testes juntamente com o cliente ("o cliente pode ver as coisas a funcionar ao longo do projeto e não apenas no dia do arranque ou formação");
    3. Realização da formação em grupos, em vez de todos os formandos ao mesmo tempo.

  Nota do investigador: todas as sugestões já se encontravam refletidas no método aquando este comentário, o que serviu como validação dos seus elementos e caraterísticas. O terceiro ponto, trata, no entanto, na forma como a formação é organizada, pelo que não deve estar contemplado no método, senão definida de acordo com o cliente e as suas necessidades (por exemplo, equipa pequena *versus* equipa grande).

Em suma, pode afirmar-se que, a partir destas reuniões e comentários, é possível reunir um conjunto de preocupações dos colaboradores Primavera, principalmente relacionado com a necessidade de comparação com o atual paradigma de implantação, onde apenas existe um *go live,* e uma grande quantidade de documentação associada. Os comentários, no entanto, representam apenas o *feedback* sobre o *roadmap* elaborado e sobre a tecnologia, não propriamente sobre o método AgileMIP, uma vez que estes elementos não tiveram disponibilidade para assistir à sua apresentação.

Em suma, as principais preocupações encontradas foram as seguintes:

- Realização da etapa de análise da MIP: na atualidade, a MIP considera, na gestão de implementação, a realização da análise, cujo objetivo é o levantamento, análise, modelação e documentação

detalhada de todo o funcionamento da organização, das suas necessidades e processos de negócio, antes de o projeto começar (Primavera BSS, 2017). No AgileMIP, a análise não ocorre dessa forma. Na etapa de iniciação, existe uma caraterização global do negócio, que não é exaustiva e não necessita de ter um elevado nível de detalhe ou estar 100% completa. Necessita, sim, de identificar os contornos globais do negócio do cliente, de acordo com as suas necessidades, uma vez que ao longo de todo o processo existe um maior espaço para conhecer o cliente e as suas especificações. Esta foi uma atividade que inicialmente não constava no método, no entanto foi adicionada como resposta à segunda reunião.

- Contabilização do esforço: não estando diretamente relacionado com o *design* do AgileMIP, esta é uma preocupação comum à maior parte dos intervenientes tanto nas reuniões, como nos comentários. Assim, no *roadmap* final elaborado para o projeto de *Eye Peak*, representado na Figura 16, foi adicionada a contabilização do esforço *on-site* e *off-site*, que permite comparar com os projetos habituais de *Eye Peak*.
- Duração do projeto/Tempo: tipicamente, na abordagem em *waterfall*, existe um planeamento de todo o trabalho a realizar num projeto. Num ciclo de vida ágil, onde se enquadra o AgileMIP, o trabalho é dividido em *sprints* de duração curta e bem delimitada no tempo, com um resultado bem definido.
- Âmbito: numa abordagem ou método tipicamente em *waterfall*, o âmbito é definido integralmente no início do projeto e permanece igual até ao seu término. No que respeita ao *agile*, esta é uma das grandes diferenças: o âmbito pode alterar-se. Isto não significa que o âmbito do projeto está constantemente em mudança, senão que deve existir consciência de que o âmbito se torna desenquadrado do objetivo da empresa, e que, por isso, pode gerar um resultado obsoleto, o que não só não deve acontecer, como não é, habitualmente, o resultado dos projetos ágeis. Assim, a gestão do âmbito, nas abordagens e modelos ágeis consiste na "identificação iterativa dos requisitos, descritos em *user stories* e escritas na linguagem que o cliente usa no seu quotidiano. O conjunto de todas as histórias forma o *Backlog*, as quais devem estar priorizadas por ordem de importância" (Sousa, 2018, p.37).
- Tempo e custo: estas variáveis fazem também parte das preocupações dos colaboradores da Primavera BSS. No entanto, similarmente ao que é praticado na MIP, na abordagem ágil também o tempo e o custo são fixos, variando apenas o âmbito (Sousa, 2018).
- Deslocações ao cliente: os consultores não querem deslocar-se várias vezes ao cliente. Assim, algumas adaptações tiveram que ser feitas, como, por exemplo, no que respeita à preparação, que

apenas ocorre uma vez. Outra consideração a ter em conta, é que as atividades não necessitam de ser realizadas presencialmente, mas também remotamente, dependendo do cliente.

- Entrega: alguns dos colaboradores continuam a defender a existência de apenas um *go live*, no final do projeto. Ora, num ciclo de vida ágil a entrega é incremental, iterativa e evolutiva (Laanti et. al., 2011; Kumar e Bhatia, 2012; Moniruzzaman e Hossain, 2013), e, também, mais rápida e frequente (Begel e Nagappan, 2007; Laanti et. al., 2011; Moniruzzaman e Hossain, 2013), resultando sempre uma versão funcional do produto no final de cada iteração (Moniruzzaman e Hossain, 2013; Papatheocharous e Andreou, 2013; Mohammed et al., 2013).
- Dependências: as dependências funcionais foram referidas por um colaborador Primavera, no caso, para o *Eye Peak*. No entanto, a rede de dependências para esta aplicação já se encontrava elaborada. A rede de dependências é, no entanto, um instrumento de elevada importância e valor, especialmente no contexto de implantação, pois permite organizar e planear a implantação de uma forma otimizada, e priorizar o trabalho a realizar, não só com base nos requisitos do cliente e nas suas necessidades, como também nas dependências funcionais da aplicação informática, que, *per si*, já apresentam uma grande complexidade.

Assim sendo, depois da incorporação de todo o *feedback* obtido das reuniões, o projeto, com data de início de 16/03/2020 e data de fim de 05/05/2020, conta com 35 dias de projeto (incluindo suporte e formação), e com cerca de 329 horas de esforço (incluindo suporte), 98 horas *on-site* e 231 horas *off-site*.

Sprint #	Atividades	Intervenientes	Nº de intervenientes	Início	Fim	On-site	Off-site	Esforço (horas)	Duração das atividades
	Iniciação			16/03/2020	20/03/2020	11,5	18,5	30,0	5 dias
-	Roadmap	GP, Consultor	2	16/03/2020	16/03/2020		3,0	3,0	1 d
-	Caraterização gobal do negócio	Consultor	1	17/03/2020	18/03/2020	7,5	7,5	15,0	2 d
-	Kick-off	GP, Cliente	1	19/03/2020	20/03/2020	4,0	8,0	12,0	1,5 d
	Sprint 1			23/03/2020	31/03/2020	15,0	28,0	43,0	7 dias
1	Análise da sprint	GP, Consultor, Cliente	2	23/03/2020	23/03/2020		4,0	4,0	02:00 h
1	Planeamento da sprint	GP, Consultor	2	23/03/2020	23/03/2020		4,0	4,0	02:00 h
1	Configuração base da aplicação	Consultor	1	24/03/2020	25/03/2020		15,0	15,0	2 d
1	Instalação e configuração dos terminais	Consultor	1	26/03/2020	27/03/2020	15,0		15,0	2 d
1	Demonstração e revisão da sprint	GP, Cliente	1	30/03/2020	30/03/2020		1,0	1,0	01:00 h
1	Formação	Formador	1	31/03/2020	31/03/2020		4,0	4,0	04:00 h
	Sprint 2			31/03/2020	09/04/2020	30,0	21,5	51,5	8 dias
2	Análise da sprint	GP, Consultor, Cliente	2	31/03/2020	31/03/2020		4,0	4,0	02:00 h
2	Planeamento da sprint	GP, Consultor	2	31/03/2020	31/03/2020		4,0	4,0	02:00 h
2	Etiquetagem	Consultor	1	01/04/2020	02/04/2020	7,5	7,5	15,0	2 d
2	Inventário	Consultor	1	03/04/2020	07/04/2020	22,5		22,5	3 d
2	Demonstração e revisão da sprint	GP, Cliente	1	08/04/2020	08/04/2020		2,0	2,0	01:00 h
2	Formação	Formador	1	09/04/2020	09/04/2020		4,0	4,0	0,5 d
	Sprint 3			09/04/2020	23/04/2020	22,5	51,5	74,0	10 dias
3	Análise da sprint	GP, Consultor, Cliente	2	09/04/2020	09/04/2020		4,0	4,0	02:00 h
3	Planeamento da sprint	GP, Consultor	2	09/04/2020	09/04/2020		4,0	4,0	02:00 h
3	Receção	Consultor	1	10/04/2020	14/04/2020	7,5	15,0	22,5	3 d
3	Expedição	Consultor	1	15/04/2020	17/04/2020	7,5	15,0	22,5	3 d
3	Arrumação	Consultor	1	20/04/2020	21/04/2020	7,5	7,5	15,0	2 d
3	Demonstração e revisão da sprint	GP, Cliente	1	22/04/2020	22/04/2020		2,0	2,0	01:00 h
3	Formação	Formador	1	23/04/2020	23/04/2020		4,0	4,0	0,5 d
	Sprint 4			23/04/2020	04/05/2020	15,0	36,5	51,5	7 dias
4	Análise da sprint	GP, Consultor, Cliente	2	23/04/2020	23/04/2020		4,0	4,0	02:00 h
4	Planeamento da sprint	GP, Consultor	2	23/04/2020	23/04/2020		4,0	4,0	02:00 h
4	Devoluções	Consultor	1	24/04/2020	27/04/2020	7,5	7,5	15,0	1,5 d
4	Reaprovisionamento	Consultor	1	28/04/2020	30/04/2020	7,5	15,0	22,5	3 d
4	Demonstração e revisão da sprint	GP, Cliente	1	01/05/2020	01/05/2020		2,0	2,0	01:00 h
4	Formação	Formador	1	04/05/2020	04/05/2020		4,0	4,0	0,5 d
	Fecho			05/05/2020	05/05/2020	4,0	0,0	4,0	1 dias
-	Post-mortem + inquérito de satisfação	GP, Consultor, Cliente	2	05/05/2020	05/05/2020	4,0		4,0	02:00 h
-	Suporte			31/03/2020	08/05/2020		75,0	75,0	10 dias

Figura 16: *Roadmap* do projeto de *Eye Peak*

Este *roadmap*, representado na Figura 16, corresponde a uma implantação *standard* do *Eye Peak*, isto é, sem customizações ou desenvolvimentos à medida. Cada projeto é único, pelo que não deve existir um planeamento *standard*, senão uma avaliação e análise individual do projeto de implantação e da organização cliente.

4.5. AgileMIP *versus* MIP

O AgileMIP preconiza a introdução de uma nova forma de trabalho para a Primavera BSS, e as suas equipas e parceiros, introduzindo uma grande mudança no método de trabalho habitualmente utilizado. Assim sendo, é inevitável percecionar e apontar as diferenças do AgileMIP face à clássica MIP, conforme apresentada na secção 4.1 deste capítulo.

4.5.1. Diferenças face à MIP

A MIP possui um ciclo de vida em *waterfall*, pelo que possui um fluxo de atividades sequencial, isto é, uma etapa segue a outra, sem retorno à anterior. Deste conceito emergem caraterísticas que identificam claramente um conjunto de diferenças face ao AgileMIP.

- **Entrega do produto, suporte, monitorização e organização do trabalho**

A forma como decorre a entrega do produto representa uma grande diferença, comparativamente com os métodos de implantação presentes na MIP (*Accelerated, Controlled* e *Extended*), e que influencia outros aspetos, tais como o suporte e a forma de monitorização e organização do trabalho.

A entrega, nestes métodos, é realizada uma única vez, no final do projeto, sendo aqui que o cliente tem contacto com o produto e o trabalho realizado, podendo fornecer o seu *feedback*. Depois da entrega, começa a fase de suporte e evolução (onde se inclui a preparação e realização do suporte e a monitorização do projeto), ou seja, estas etapas apenas ocorrem depois da disponibilização do produto.

O AgileMIP não possui as caraterísticas anteriormente referidas. Principalmente, porque preconiza um ciclo de vida ágil, no qual não se enquadra um fluxo de atividades sequencial. É iterativo e incremental, onde se valoriza a participação e o contínuo envolvimento do cliente. Assim sendo, o trabalho é organizado por *sprint*s, de onde resulta sempre a entrega de um incremento funcional ao cliente, o qual poderá ser imediatamente utilizado pelo cliente, sempre que aplicável.

Esta caraterística requer que o suporte ocorra desde cedo no projeto, nomeadamente desde a primeira entrega realizada ao cliente. Isto representa uma grande diferença no processo de implantação Primavera, uma vez que o suporte é visualizado como um novo projeto, que se inicia apenas no final da execução do projeto.

Também a monitorização ocorre de forma distinta da MIP, ocorrendo também num *continuum,* ao longo de todo o projeto.

- **Etapa de análise e documento de visão de processos**

Na MIP a etapa de análise consiste no levantamento, análise, modelação e documentação detalhada do funcionamento da organização, os seus processos de negócio, necessidades, e do setor de atividade onde se enquadra (Primavera BSS, 2017). Isto significa que o conhecimento da organização é realizado, detalhado e descrito como um todo antes do projeto começar.

Esta informação é descrita no documento de visão de processos, onde constam todos os processos que serão tratados no projeto de implantação. Para além da descrição de cada processo, são também elaborados diagramas de alto nível, com *Unified Modeling Language* (UML) (Primavera BSS, 2017).

No AgileMIP esta atividade é realizada de forma diferente, uma vez que existe um maior espaço para interagir e conhecer o cliente ao longo do projeto. Na etapa de preparação ocorre a caraterização global do negócio, onde é procurado identificar os processos de negócio mais críticos do cliente, que serão mais afetados pelo processo de implantação, bem como conhecer o cliente e os seus colaboradores, a sua opinião, dificuldades e os motivos que levaram à necessidade de mudança e inovação. Esta informação é descrita no documento de visão global do projeto, que é atualizado ao longo do decorrer do projeto.

- **Levantamento de requisitos**

O levantamento de requisitos, na MIP, é realizado uma vez no projeto, na fase de venda, mais concretamente na etapa de descoberta (Primavera BSS, 2017). Isto significa que os requisitos são identificados *a priori*, antes do projeto começar.

No AgileMIP o levantamento de requisitos ocorre sob a forma de *user stories*, durante a fase de pré-projeto e projeto. Isto significa que, a qualquer momento do projeto, podem ser atualizadas, adicionadas ou removidas *user stories*.

- **Gestão do sucesso**

Na atual MIP, não existe nenhuma referência à gestão do sucesso, pelo que este é um elemento diferenciador do AgileMIP.

- **Separação da gestão de projeto da implantação**

Na MIP existe uma distinção entre o que é "gestão da implementação" e gestão do projeto, sendo descritas como atividades distintas, com procedimentos e objetivos distintos. A "gestão da implementação", procura efetuar a "implementação" da solução pretendida pelo cliente em ambiente de produção, enquanto a gestão de projeto visa gerir o projeto para que este alcance o sucesso (Primavera BSS, 2017). Distinguem-se também relativamente às suas etapas: gestão da implementação (venda, execução e suporte e evolução), e gestão de projetos (planeamento, monitorização e controlo e fecho).

O AgileMIP contempla uma visão diferente, de implantação. A implantação consiste num processo que contempla atividades de origem técnica e organizacional, e cujo objetivo é melhorar a organização e os seus processos de negócio.

- **Priorização do trabalho**

Conforme já referido anteriormente, a MIP possui um fluxo de atividades sequencial, tipicamente em *waterfall*, com apenas uma entrega no final do projeto. Por esse motivo, não existe a necessidade de priorizar o trabalho, uma vez que será entregue de uma vez só ao cliente.

No AgileMIP a entrega é incremental e existe uma grande abertura à mudança. O que significa que, a qualquer momento, podem ser introduzidas novas necessidades ou alterações de prioridades do cliente relativamente aos requisitos. Assim, é necessário priorizar continuamente o trabalho de acordo com as necessidades do cliente, garantindo, desta forma, que cada entrega representa o máximo valor para o cliente.

- **Redesenho dos processos de negócio**

O redesenho dos processos de negócio é uma atividade que, apesar de não descrita na MIP, é realizada na prática. No AgileMIP esta atividade é contemplada na etapa de realização e preconiza uma atividade de gestão de processos de negócio, e, consequentemente, de desenvolvimento organizacional. É realizada iterativamente ao longo do projeto, uma vez que é uma das atividades das *sprints*. Isto significa que esta atividade, que na maior parte dos casos afeta as organizações e o seu funcionamento, terá menor impacto, uma vez que a mudança dos processos de negócio não ocorre de uma vez só, conforme aconteceria num fluxo tipicamente em *waterfall*.

Esta atividade é também "suavizada" por outros elementos do AgileMIP, tal como as operações (suporte e monitorização), e a gestão da mudança. Com isto, garante-se não só uma correta transição para o novo processo a implantar, como também a correta adaptabilidade da organização aos novos processos de negócio, e uma maior aceitação destes novos processos pelas pessoas da organização, uma vez que se trata de um processo gradual com acompanhamento contínuo.

- **Formação**

Na MIP, a atividade de formação é realizada no final do projeto, na etapa de preparação final, ou seja, depois de aprovado e realizado o trabalho.

No AgileMIP a formação pode ser realizada em cada iteração (na etapa de disponibilização), depois de o trabalho realizado em cada *sprint* ser aprovado pelo cliente (na revisão da *sprint*). Assim, promove-se que cada incremento será corretamente utilizado pelo cliente no seu contexto de produção.

- **Etapa de venda:**

A etapa de venda é bastante similar entre o AgileMIP e a MIP, distinguindo-se maioritariamente pelas nomenclaturas atribuídas e pela atividade de fecho. Tanto o AgileMIP como a MIP contemplam atividades de contacto, avaliação e descoberta do negócio, e de discussão ou argumentação. Também os *deliverables* são bastante similares, coexistindo elementos como a proposta e contrato comercial. Na MIP é também contemplada uma *checklist* de análise funcional e não funcional, cujo objetivo é auxiliar na tarefa de levantamento de requisitos, sendo estes aqui registados. Para além deste documento, contempla também um documento de qualificação de negócio (Primavera BSS, 2017).

No AgileMIP, os *users stories* começam também a ser especificados nesta etapa, sendo registados no *backlog*, que é atualizado ao longo do projeto. Cada *user story* tem tipicamente esta estrutura: Como <papel do utilizador> quero <objetivo> para <benefício> (Rubin, 2012).

Outra diferença é a atividade de fecho da venda. Na MIP a etapa de venda é fechada, uma vez que se chega a um consenso do que será feito no projeto, e, a partir desse momento, tipicamente não existem alterações ao âmbito do projeto. É, no entanto, contemplada a possibilidade de aparecimento de novas vendas no decorrer do projeto.

No caso do AgileMIP não existe um fecho definitivo da etapa de venda, sendo esta atividade um *continuum*, conforme representado na Figura 14. Neste método de implantação, o âmbito pode ser modificado ou ajustado, e podem surgir novos requisitos e necessidades que originem novas vendas e projetos ou que influenciem o projeto de forma significativa. Não obstante, quase inevitavelmente, terá de haver uma venda inicial, que poderá evoluir caso seja vontade do cliente, em concordância com a Primavera.

4.5.2. Similaridades com a MIP

Apesar das diferenças apontadas na secção anterior (5.4.1) existem também caraterísticas comuns entre a MIP e o AgileMIP, que são:

- **Atividade e documento de *kick-off:***

Esta atividade é contemplada tanto na MIP, como no AgileMIP, com o mesmo objetivo em ambos os elementos: apresentar as equipas envolvidas no projeto (de implantação e do cliente) e marcar o início formal do projeto. O documento de *kick-off*, também considerado em ambas as abordagens, consiste no suporte digital para a reunião.

- **Pedidos de mudança**

Tanto a MIP, como o AgileMIP contemplam os pedidos de mudança. No entanto, na MIP este pedido é elaborado sempre que quaisquer alterações sejam feitas aos requisitos, plano do projeto, âmbito,

esforço ou critérios de qualidade previstos no plano inicial (Primavera BSS, 2017). Neste documento, descreve-se a alteração, avalia-se o seu impacto (nos requisitos, no projeto, o que será impactado, eliminado ou produzido), e regista-se o conhecimento e aprovação do cliente (Primavera BSS, 2017).

No AgileMIP, o documento não é realizado com tanta regularidade, uma vez que é um método ágil, aberto à mudança. Por outras palavras, a mudança é acomodada de forma "natural", aquando do planeamento de cada *sprint*. Nos projetos que sigam este método de implantação deve ser sempre contemplada uma margem para mudanças e desvios, em termos de custo e tempo. Sempre que surja uma mudança, alteração, uma nova necessidade ou uma ideia de importância para o cliente, o trabalho deve ser priorizado.

Caso a mudança, alteração, necessidade ou ideia preconize um grande impacto no projeto, no seu custo ou duração, o cliente deverá ser contactado e devem ser expostas estas questões. Primeiramente, deverá ser analisado, conjuntamente, o *backlog* e percecionado se, de facto, existem elementos que possam ser substituídos e que têm menos prioridade (trabalho realizado na validação do *backlog*). Se não for possível abdicar de nenhum elemento, deverá projetar-se uma nova venda ao cliente.

Caso este não esteja disponível para esta nova venda, o projeto é realizado na sua normalidade, de acordo com a priorização estabelecida (continuamente atualizada), e, no final do projeto ou *sprint*, são demonstradas as *user stories* que não foram concretizadas e o porquê de tal ter sucedido (na retrospetiva da *sprint* ou na reunião de encerramento do projeto).

- Gestão da mudança:

A gestão da mudança é contemplada tanto no AgileMIP, como na MIP. O objetivo da gestão da mudança é apoiar a organização cliente e os seus colaboradores na implantação da nova aplicação informática, garantindo uma maior aceitação e adaptação às novas práticas e, consequentemente, o sucesso do projeto.

- Lista de presenças:

A lista de presenças é um elemento presente tanto na MIP, como no AgileMIP. O seu objetivo é comprovar a realização de uma formação, quem esteve presente e quem foi a pessoa responsável.

- Reunião e documento de fecho do projeto

A reunião e o documento de fecho são contemplados tanto no AgileMIP, como na MIP. O objetivo é fazer uma síntese do trabalho realizado no projeto, identificar o que ficou por fazer e porquê, o que poderia ter corrido melhor e efetuar recomendações (Primavera BSS, 2017). Nesta reunião estão presentes tanto a equipa do cliente, como a equipa Primavera.

5 Conclusão

Do trabalho realizado, conclui-se que o *agile* é ainda pouco aplicado no contexto de implantação de TI, conforme pode ser constatado através da análise dos métodos de implantação elaborada. No entanto, pode concluir-se que a sua aplicabilidade neste contexto é válida e exequível, sempre tendo em atenção as caraterísticas da organização, dos seus projetos e das equipas.

Para o desenvolvimento da dissertação foram adotadas as orientações do *Design Science Research*, tendo sido seguidas as seis etapas que o compõe. Primeiramente, foi realizada uma revisão da literatura, que permitiu conhecer o problema e as suas especificidades, nomeadamente dos ciclos de vida dos projetos, do *agile* e do contexto de implantação. Aqui, foram também analisados e classificados diferentes métodos de implantação de TI. Depois de obtido o conhecimento necessário sobre o problema e de clarificada a motivação do trabalho, foram estabelecidos os objetivos. De seguida, foram recolhidas as caraterísticas desejadas para o novo artefacto (um novo método de implantação de TI), através de uma análise detalhada do contexto organizacional da Primavera BSS, da análise e exploração dos princípios e valores ágeis e, também, dos métodos de implantação com um ciclo de vida ágil. Finalmente, procedeu-se a criação do artefacto, à sua demonstração (apresentação às equipas Primavera) e avaliação (reflexão sobre o *feedback* obtido na etapa anterior), tendo sido realizadas quatro iterações. A comunicação dos resultados obtidos ocorreu através da presente dissertação e da redação de vários artigos científicos.

O artefacto criado é um novo método de implantação de TI (AgileMIP) que tem como objetivo trazer maior agilidade aos projetos de implantação da Primavera BSS, incorporando as boas práticas do mercado, respondendo às dificuldades no fecho dos projetos de implantação Primavera, ao contexto de inovação tecnológica vivenciado através do lançamento do ERP Rose, bem como aos diversos pedidos de clientes e parceiros relativamente à existência de um método ágil de implantação.

Os principais contributos desta dissertação consistem na clarificação das principais caraterísticas do *agile* no contexto de implantação de TI, e na proposta de um método ágil de implantação de TI adequado às necessidades e contexto organizacional da Primavera BSS, que possibilita um maior envolvimento do cliente ao longo de todo o processo de implantação, maior transparência sobre o projeto, entregas frequentes, incrementais e contínuas, a entrega de um produto com maior qualidade e que vai de encontro às reais necessidades do cliente, e que suporta os processos organizacionais da empresa cliente.

As principais dificuldades sentidas no decorrer do trabalho advêm, principalmente, da falta de uniformidade da linguagem utilizada na literatura, sendo frequente os autores utilizarem o mesmo termo

indiferenciadamente referindo-se a objetos de interesse e a contextos diferentes, sendo alguns exemplos método, modelo, processo, implementação e implantação, e TI e SI. Outra dificuldade foi encontrar métodos de implantação de TI e a sua descrição, uma vez que as organizações não partilham abertamente os seus métodos, por acreditarem que estes traduzem vantagens competitivas.

A avaliação do AgileMIP consiste também numa limitação da presente dissertação, influenciada pela conjuntura de pandemia, onde se realizou este trabalho. Assim, impossibilitou a realização de um projeto piloto que comprovasse empiricamente a aplicabilidade e os benefícios do AgileMIP.

Como trabalho futuro, sugere-se a aplicação e validação do AgileMIP em vários projetos de implantação Primavera, para que seja comprovada a sua aplicabilidade e os seus benefícios, e, também, para que sejam efetuadas as melhorias necessárias. De seguida, sugere-se a criação de uma formação sobre o *agile* e o AgileMIP, para colaboradores e parceiros Primavera, bem como a abertura da MIP à comunidade.

Por fim, sugere-se a realização de mais estudos empíricos que comprovem as forças e limitações dos diferentes ciclos de vida e métodos analisados nesta dissertação, bem como uma maior exploração do *agile* no contexto de implantação, algo que está ainda em desenvolvimento.

Referências

Accenture. (2014). *Accenture CAS: Solution Implementation—Making change happen*. Accenture. https://www.accenture.com/t20150728t142612__w__/us-en/_acnmedia/accenture/conversion-assets/microsites/documents22/brochure_accenture-cas_solution-implementation_web.pdf

Advanced Solutions International, A. (2019). *ASI Client Sucess Agile Implementation Methodology*. https://www.advsol.com/ASI/SupportPortal/Consulting/Agile_Implementation_Methodology.aspx

Agile Alliance, & PMI, P. M. I. (2017). *Agile Practice Guide*. Project Management Institute, Inc. https://www.agilealliance.org/wp-content/uploads/2017/09/AgilePracticeGuide.pdf

Agrawal, A., Atiq, M. A., & Maurya, L. S. (2016). *A Current Study on the Limitations of Agile Methods in Industry Using Secure Google Forms*. *78*, 291–297. Scopus. https://doi.org/10.1016/j.procs.2016.02.056

Alshamrani, A., & Bahattab, A. (2015). A comparison between three SDLC models waterfall model, spiral model, and incremental/iterative model. *International Journal of Computer Science Issues*, *12*(1), 106–111. https://www.ijcsi.org/papers/IJCSI-12-1-1-106-111.pdf

Amlani, R. D. (2012). Advantages and Limitations of Different SDLC Models. *International Journal of Computer Applications & Information Technology*, *1*(3), 6–11. http://www.ijcait.com/IJCAIT/13/1334.pdf

Anwar, A. (2014). A review of RUP (Rational Unified Process). *International Journal of Software Engineering*, *5*(2), 8–24. https://www.cscjournals.org/manuscript/Journals/IJSE/Volume5/Issue2/IJSE-142.pdf

Arlow, J., & Neustadt, I. (2002). *UML and the Unified Process*. Addison-Wesley. https://www.academia.edu/8751505/UML_-_and_the_Unified_Process

Avison, D. E., & Fitzgerald, G. (2006). Methodologies for Developing Information Systems: A historical perspective. In *The past and future of Information Systems: 1976-2006 and beyond:* (p. 27–38). https://www.researchgate.net/publication/227101490_Methodologies_for_Developing_Information_Systems_A_Historical_Perspective

Axelos. (2018). PRINCE2 Agile Guidance Preview. In *PRINCE2 Agile Guidance* (p. 21). The Stationary Office. https://www.axelos.com/store/book/prince2-agile

Azanha, A., Tiradentes Terra Argoud, A. R., de Camargo Junior, J. B., & Antoniolli, P. D. (2017). Agile project management with Scrum A case study of a Brazilian pharmaceutical company IT project. *International Journal of Managing Projects in Business*, *10*(1), 121–142. https://doi.org/10.1108/IJMPB-06-2016-0054

Babic, V. (2010). *The Agile Sure Step Project Type: New Options for Implementing Microsoft Dynamics | MSDynamicsWorld.com*. https://msdynamicsworld.com/story/dynamics-ax/agile-sure-step-project-type-new-options-implementing-microsoft-dynamics

Balaji, S., & Murugaiyan, M. S. (2012). Wateerfall Vs V-Model Vs Agile: A compative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), 26–30. https://www.academia.edu/11483288/COMPARATIVE_STUDY_WATERFALL_V_S_AGILE_MODEL

Beck, K. et. al. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/

Begel, A. e Nagappan, N. (2007). *Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study*. First International Symposium on Empirical Software Engineering and Metrics. https://www.microsoft.com/en-us/research/publication/usage-and-perceptions-of-agile-software-development-in-an-industrial-context-an-exploratory-study/

Benbya, H. e McKelvey, B. (2006). Toward a complexity theory of information systems development. *Information Technology & People*, *19*(1), 12–34. https://doi.org/10.1108/09593840610649952

Beynon-Davies, P. (2016). *Information Systems Development: An Introduction to Information Systems Engineering* (3ª). Macmillan International Higher Education. https://books.google.pt/books?hl=en&lr=&id=8jtdDwAAQBAJ&oi=fnd&pg=PR9&dq=%22information+systems+development%22&ots=1zTUfNai0V&sig=0p90a9hvI1mRKeHCvxAJxPVAHZo&redir_esc=y#v=onepage&q&f=false

Bhuvaneswari, T., & Prabaharan, S. (2013). A survey on Software Development Life Cycle Models. *Journal of Computer Science and Information Technology*, *2*(5), 262–267. http://d.researchbib.com/f/2nq3q3YzydL3AgLI5wo20iMT9wpl9jLKOypaZiGJS5ZwNkZl9JZxx1ZwNkZmt0YaOxMt.pdf

Bishop, D., Rowland, P., & Noteboom, C. (2018). Antecedents of Preference for Agile Methods: A Project Manager Perspective. *Proceedings of the 51st Hawaii International Conference on System Sciences*, 5435–5444. https://aisel.aisnet.org/hicss-51/st/agile_development/3

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61–72. https://doi.org/10.1109/2.59

Braun, C., Wortmann, F., Hafner, M., & Winter, R. (2005). Method Construction—A Core Approach to Organizational Engineering. *Proceedings of the 2005 ACM Symposium on Applied Computing*, 1295–1299. https://doi.org/10.1145/1066677.1066971

Brinkkemper, S. (1996). Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, *38*(4), 275–280. https://doi.org/10.1016/0950-5849(95)01059-9

Carugati, A. (2008). Information system development activities and inquiring systems: An integrating framework. *European Journal of Information Systems - EJIS*, *17*, 143–155. https://doi.org/10.1057/ejis.2008.5

Carvalho, J. A. (2000). Information System? Which One Do You Mean? In E. D. Falkenberg, K. Lyytinen, & A. A. Verrijn-Stuart (Eds.), *Information System Concepts: An Integrated Discipline Emerging* (p. 259–277). Springer US. https://doi.org/10.1007/978-0-387-35500-9_22

Carvalho, L. G. (2018). *Já ouviram falar no PRINCE2 AGILE®?* https://www.linkedin.com/pulse/j%C3%A1-ouviram-falar-prince2-agile-luiz-guilherme-carvalho/

Carvalho, M. A., & Brito, S. S. (2017). Organizational Change. *Superavit*, *2*, 89–99. https://doi.org/10.26358/srgivol2ar26

Cherry Road Technologies. (n.d.). *Cherry Road Oracle Cloud Implementation Methodology*. Cherry Road Technologies. https://www.cherryroad.com/wp-content/uploads/2019/04/CherryRoad-Oracle-Cloud-Implementation-Methodology.pdf

Cockburn, A. (2016). The heart of agile. *The Journal of Defense Software Engineering*, *29*(6), 4–6. http://static1.1.sqspcdn.com/static/f/702523/27327106/1478725625433/201611-0-Issue.pdf?token=xgsyaiChJlRLn3tFqol3%2FkXMGPU%3D

CollabNet. (2019). *VersionOne Implementation Services*. CollabNet VersionOne Resources. https://resources.collab.net/versionone-datasheets/versionone-implementation-services

CollabNet, & VersionOne. (2019). *13th Annual State of Agile Survey* (Anual No. 13; p. 16). https://www.stateofagile.com/home

Cooke, J. L. (2016). *PRINCE2 Agile An Implementation Pocket Guide: Step-by-step advice for every project type*. IT Governance Ltd. https://books.google.pt/books?hl=en&lr=&id=FOA2DwAAQBAJ&oi=fnd&pg=PA1&dq=%22prince2+agile%22&ots=qHXuED5A8b&sig=po7V0vpFTkRDbPFbhHXpUTHpbh0&redir_esc=y#v=onepage&q=%22prince2%20agile%22&f=false

D'Ascenção, L. C. M. (2001). *Organização, sistemas e métodos: Análise, redesenho e informatização de processos administrativos* (1st ed.). Atlas.

Davenport, T., & Short, J. (1990). The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review*, *31*(4), 11–27. https://sloanreview.mit.edu/article/the-new-industrial-engineering-information-technology-and-business-process-redesign/

Deloitte. (2020). Agile Implementation & Development. *Deloitte Digital Austria*. https://www.deloittedigital.at/en/offerings/agile-implementation-development/

Eason, O. K. (2016). *Information Systems Development Methodologies Transitions: An Analysis of Waterfall to Agile Methodology* [Honors Theses and Capstones, University of New Hampshire]. https://scholars.unh.edu/honors/286

EU, P. O. of the E. U. (2016). *PM² project management methodology guide: Open edition.* (Open Edition, v.0.0.9). European Commission, DIGIT Centre of Excellence in Project Management. http://op.europa.eu/en/publication-detail/-/publication/0e3b4e84-b6cc-11e6-9e3c-01aa75ed71a1

Fernandez, D. J., & Fernandez, J. D. (2008). Agile Project Management—Agilism versus Traditional Approaches. *Journal of Computer Information Systems*, *49*(2), 10–17. https://doi.org/10.1080/08874417.2009.11646044

Fitzgerald, B. (2000). Systems development methodologies: The problem of tenses. *Information Technology & People*, *13*(3), 174–185. https://doi.org/10.1108/09593840010377617

Fitzgerald, B., Russo, N., & O'Kane, T. (2000). An Empirical Study of System Development Method Tailoring in Practice. *ECIS 2000 Proceedings*, *4*. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1045&context=ecis2000

Forte, F., & Kloppenborg, T. (2017). *The Agile Mindset for Project Management*. 1–15. https://doi.org/10.5130/pmrp.irnop2017.5740

Graycell. (n.d.). *Graycell PeopleSoft Implementation*. Retrieved 30 April 2020, from http://www.graycellamerica.com/implementation_peoplesoft.htm

Gulledge, T., & Simon, G. (2005). The evolution of SAP implementation environments: A case study from a complex public sector project. *Industrial Management & Data Systems*, *105*(6), 714–736. https://doi.org/10.1108/02635570510606969

Gustavsson, T. (2016). *Benefits of Agile Project Management in a Non-Software Development Context: A Literature Review*. 114–124. http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-70311

Harmsen, F., Brinkkemper, S., & Oei, J. L. H. (1994). Situational method engineering for informational system project approaches. *Proceedings of the IFIP WG 8.1 Working Conference*, 169–174. https://dl.acm.org/citation.cfm?id=717382

Henderson-Sellers, B. (2006). Method Engineering: Theory and Practice. *Information Systems Technology and Its Applications*, 13–23. https://www.researchgate.net/publication/221141428_Method_Engineering_Theory_and_Practice

Highsmith, J. A., & Highsmith, J. (2002). *Agile Software Development Ecosystems*. Addison-Wesley Professional. https://books.google.pt/books/about/Agile_Software_Development_Ecosystems.html?id=0aDjuldhtCwC&redir_esc=y

IBM. (n.d.). *IBM Cognos Solutions Implementation Methodology: IBM Cognos 8 Controller Implementation Roadmap—Structured Steps, Proven Practices, Enterprise Success*. Cognos Software. https://docuri.com/download/fs-ibm-cognos-solutions-implementation-methodology_59ae46fff581710a6200d03e_pdf

IBM Software. (2011). *IBM Business Analytics Solutions Implementation Method (BASIM)*. IBM Corporation. ftp://public.dhe.ibm.com/software/data/sw-library/cognos/services/pdfs/implementation/business_analytics_solutions_implementation_method_data_sheet.pdf

IFS. (2019). *An explanation of the IFS Implementation Methodology™*. https://www.ifs.com/corp/sitecore/media-library/assets/2018/08/17/ifs-implementation-methodology/

Iivari, J., & Venable, J. (2009). *Action research and design science research—Seemingly similar but decisively dissimilar*. 1642–1653. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1025&context=ecis2009

Infor Project Management Office. (2014). *Infor Deployment Method*. Infor. https://www.infor.com/content/whitepapers/deployment-method.pdf/

Isetta, S., & Sampietro, M. (2018). Agile in ERP Projects. *PM World Journal*, VII(IX), 1–18. https://pmworldlibrary.net/wp-content/uploads/2018/09/pmwj74-Sep2018-Isetta-Sampietro-agile-in-erp-projects-featured-paper.pdf

Kamei, F., Pinto, G., Cartaxo, B., & Vasconcelos, A. (2017). On the Benefits/Limitations of Agile Software Development. *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 154–159. https://dl.acm.org/doi/abs/10.1145/3084226.3084278

Korpela, M., Mursu, A., & Soriyan, H. A. (2002). Information Systems Development as an Activity. *Computer Supported Cooperative Work (CSCW)*, *11*(1), 111–128. https://doi.org/10.1023/A:1015252806306

Kronos. (2019). *Kronos Paragon Brochure: Fact Sheet*. Kronos. https://www.kronos.com/resources/kronos-paragon-brochure

Kruchten, P. (2001). *The Rational Unified Process: An introduction* (3rd ed.). Addison-Wesley. https://www.researchgate.net/publication/220018149_The_Rational_Unified_Process–An_Introduction

Kumar, G., & Bhatia, P. K. (2012). Impact of Agile Methodology on Software Development Process. *International Journal of Computer Technology and Electronics Engineering*, *2*(4), 46–50. https://www.researchgate.net/publication/255707851_Impact_of_Agile_Methodology_on_Software_Development_Process

Laanti, M., Salo, O., & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology*, *53*(3), 276–290. https://doi.org/10.1016/j.infsof.2010.11.010

Law, E. L.-C., & Lárusdóttir, M. K. (2015). Whose Experience Do We Care About? Analysis of the Fitness of Scrum and Kanban to User Experience. *International Journal of Human-Computer Interaction*, *31*(9), 584–602. Scopus. https://doi.org/10.1080/10447318.2015.1065693

Layton, M. C., & Ostermiller, S. J. (2017). *Agile Project Management For Dummies*. John Wiley & Sons. https://books.google.pt/books/about/Project_Management_Checklists_For_Dummie.html?id=WuThBQAAQBAJ&source=kp_book_description&redir_esc=y

LLP Group. (2019). *Methods*. LLP International - SunSystems Consulting. https://llpinternational.com/methods/

Machado, L. C. P., & Neiva, E. R. (2017). Práticas de gestão da mudança: Impacto nas atitudes e nos resultados percebidos. *Revista Psicologia Organizações e Trabalho*, *17*(1), 22–29. https://doi.org/10.17652/rpot/2017.1.12157

Mahalakshmi, M., & Sundararajan, M. (2013). Traditional SDLC Vs Scrum Methodology—A Comparative Study. *International Journal of Emerging Technology and Advanced Engineering*, *3*(6), 192–196. https://pdfs.semanticscholar.org/7740/829e70c028a75780d3b7bd034345beb940c4.pdf

Matkovic, P., & Tumbas, P. (2010). A Comparative Overview of the Evolution of Software Development Models. *Journal of Industrial Engineering and Management*, *1*(4), 163–172. https://www.researchgate.net/publication/267711880_A_Comparative_Overview_of_the_Evolution_of_Software_Development_Models

Maurer, T. J. (2001). Career-relevant learning and development, worker age, and beliefs about self-efficacy for development. *Journal of Management*, *27*(2), 123–140. https://doi.org/10.1016/S0149-2063(00)00092-1

McConnell, S. (1996). *Rapid Development* (1 edition). Microsoft Press. https://mycomputersciencebooks.files.wordpress.com/2017/08/steve-mcconnell-rapid-development-taming-wild-software-schedules.pdf

McCormick, M. (2012). *Waterfall vs. Agile Methodology*. MPCS Inc. http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf

Mohammed, A. H., Alwada'n, T., & Ababneh, J. (2013). Agile Software Methodologies: Strength and Weakness. *International Journal of Computer Science and Technology*, *5*(3), 455–459. https://www.semanticscholar.org/paper/Agile-Software-Methodologies%3A-Strength-and-Weakness-Mohammad-Alwada/3accf2a39e3532e8ed9b592e7671c14f6dc4c7fe

Moniruzzaman, A. B. M., & Hossain, D. S. A. (2013). Comparative Study on Agile software development methodologies. *Global Journal of Computer Science and Technology*, *13*(7). http://arxiv.org/abs/1307.3356

Moran, J. W., & Brightman, B. K. (2000). Leading organizational change. *Journal of Workplace Learning*, *12*(2), 66–74. https://doi.org/10.1108/13665620010316226

Narayana, L. K. (2019). *ServiceNow Implementation—Agile (Scrum) Project Management Methodology | LinkedIn*. https://www.linkedin.com/pulse/servicenow-implementation-agile-scrum-project-lokesh-kumar-narayana/

Obrutsky, S. (2016). *Comparison and contrast of project management methodologies PMBOK and SCRUM*. https://www.researchgate.net/publication/305969672_Comparison_and_contrast_of_project_management_methodologies_PMBOK_and_SCRUM

Odell, J. J. (1996). A Primer to Method Engineering. In S. Brinkkemper, K. Lyytinen, & R. J. Welke (Eds.), *Method Engineering: Principles of method construction and tool support* (pp. 1–7). Springer US. https://doi.org/10.1007/978-0-387-35080-6_1

Odoo. (n.d.). *Odoo Online Implementation—Odoo 9.0 documentation*. Odoo Online Implementation. Retrieved 8 January 2020, from https://www.odoo.com/documentation/user/9.0/getting_started/documentation.html

Osis, J., & Donins, U. (2017). *Topological UML Modeling: An Improved Approach for Domain Modeling and Software Development* (1st ed.). Elsevier. https://www.researchgate.net/publication/320709806_Topological_UML_Modeling_An_Improved_Approach_for_Domain_Modeling_and_Software_Development

Osorio, J. A., Chaudron, M. R. V., & Heijstek, W. (2011). Moving from Waterfall to Iterative Development: An Empirical Evaluation of Advantages, Disadvantages and Risks of RUP. *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, 453–460. https://doi.org/10.1109/SEAA.2011.69

Overhage, S., & Schlauderer, S. (2012). Investigating the Long-Term Acceptance of Agile Methodologies: An Empirical Study of Developer Perceptions in Scrum Projects. *2012 45th Hawaii International Conference on System Sciences*, 5452–5461. https://doi.org/10.1109/HICSS.2012.387

Papatheocharous, E., & Andreou, A. S. (2013). Evidence of Agile Adoption in Software Organizations: An Empirical Survey. In F. McCaffery, R. V. O'Connor, & R. Messnarz (Eds.), *Systems, Software and Services Process Improvement* (pp. 237–246). Springer. https://doi.org/10.1007/978-3-642-39179-8_21

Pee, L. G., Kankanhalli, A., & Kim, H.-W. (2010). Knowledge Sharing in Information Systems Development: A Social Interdependence Perspective. *Journal of the Association for Information Systems*, *11*(10). https://doi.org/10.17705/1jais.00238

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

PMI, Project Management Institute. (2013). *Managing Change in Organizations: A Practice Guide* (None edition). Project Management Institute.

PMI, Project Management Institute. (2017). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)* (6th ed.). Project Management Institute, Inc.

Primavera BSS. (n.d.). *PKB: Primavera Knowledge Base*. PKB. https://www.primaverabss.com/pkb/Homepage-Login.aspx

Primavera BSS. (2017). *MIP: Metodologia de Implementação Primavera*. MIP: Projects in Motion. https://mip.primaverabss.com/

Primavera BSS. (2020). *Primavera: Coded for people*. https://pt.primaverabss.com/pt/primavera/

Ralyté, J., Rolland, C., & Deneckère, R. (2004). Towards a Meta-tool for Change-Centric Method Engineering: A Typology of Generic Operators. In A. Persson & J. Stirna (Eds.), *Advanced Information Systems Engineering* (p. 202–218). Springer. https://doi.org/10.1007/978-3-540-25975-6_16

Rational Software. (2011). *Rational Unified Process: Best Practices for Software Development Teams* (White Paper No. TP026B; p. 21). Rational Software. https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Reascos, I., & Carvalho, J. A. (2018). A Conceptual Framework for the Implantation of Enterprise Applications in Small and Medium Enterprises (SMEs). In Á. Rocha & T. Guarda (Eds.), *Proceedings of the International Conference on Information Technology & Systems (ICITS 2018)* (p. 50–61). Springer International Publishing. https://doi.org/10.1007/978-3-319-73450-7_6

Reascos Paredes, I., & Carvalho, J. (2019). SImplE: A Framework for the Successful Implantation of Enterprise IT Applications in Small and Medium Enterprises. *International Conference on Information Systems Development (ISD)*. https://aisel.aisnet.org/isd2014/proceedings2019/ISDMethodologies/15

Roque, R. P. (1998). *Estudo comparativo de metodologias de desenvolvimento de sistemas de informação utilizando a técnica Delphi /* [Dissertação de mestrado, Universidade Federal de Santa Catarina]. https://repositorio.ufsc.br/handle/123456789/77681

Royce, W. W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques. *Proceedings of the 9th International Conference on Software Engineering*, 328–338. http://dl.acm.org/citation.cfm?id=41765.41801

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley. http://ce.sharif.edu/courses/97-98/2/ce418-1/resources/root/Books/[Addison-Wesley%20Signature%20Series%20(Cohn)]%20Kenneth%20S.%20Rubin%20-%20Essential%20Scrum_%20A%20Practical%20Guide%20to%20the%20Most%20Popular%20Agile%20Process%20(2012,%20Addison-Wesley%20Professional).pdf

SABA. (2014). *Lean and Agile Project Methodology*. https://www.slideshare.net/SabaSoftware/wp-expertise-work

SAP. (2018). *SAP Activate Methodology: Overview*. https://cdn2.hubspot.net/hubfs/2605195/Whitepapers,%20Checklisten%20zu%20verschiedenen%20Themen/SAP%20Activate/SAP%20Activate%20Methodik.pdf

Sarker, I. H., Faruque, F., Hossen, U., & Rahman, A. (2015). A survey of software development process models in software engineering. *ResearchGate*, *9*(11), 55–70. https://www.researchgate.net/publication/298656663_A_survey_of_software_development_process_models_in_software_engineering

Schwaber, K. (2009). *Scrum guide*. Scrum Alliance. https://www.qagile.pl/wp-content/uploads/2017/01/Scrum-Guide-May-2009.pdf

Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide*. https://www.scrumguides.org/scrum-guide.html

Serour, M. K., & Henderson-Sellers, B. (2004). Introducing agility: A case study of situational method engineering using the OPEN process framework. *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, *1*, 50–57. https://doi.org/10.1109/CMPSAC.2004.1342805

ServiceNow. (2018). *ServiceNow Implementation Methodology: Data Sheet*. ServiceNow, Inc. https://www.servicenow.com/content/dam/servicenow-assets/public/en-us/doc-type/data-sheet/ds-sim.pdf

Shelly, G., Cashman, T. J., & Rosenblatt, H. J. (2007). *Systems Analysis and Design* (7th ed.). Cengage Learning. https://books.google.pt/books?id=i_7EnA2cS5sC&pg=PA484&lpg=PA484&dq=implementation+pilot+and+parallel&source=bl&ots=W2UCHDY-d&sig=ACfU3U18QXn5BDpf_gveat7IThOPuZ8Ang&hl=en&sa=X&ved=2ahUKEwjryeaWuavmAhWH2BQKHT8LCiI4ChDoATARegQICRAB#v=onepage&q=implementation%20pilot%20and%20parallel&f=false

Sidky, A. (2015). *The Secret to Achieving Sustainable Agility at Scale* [Business]. AgileNZ Conference, New Zealand. https://www.slideshare.net/AgileNZ/ahmed-sidky-keynote-agilenz

Silva, C. G. T., & Neto, M. T. R. (2019). Comportamento organizacional e a sua relação com a implantação de um ERP: Uma revisão. *Revista da Faculdade de Administração e Economia*, *9*(2), 80–93. https://doi.org/10.15603/2176-9583/refae.v9n2p80-93

Silveira, M., & Diniz, E. (2002). The relation between organizational change and Information Systems deployment: A study in auto parts industry. *ResearchGate*, *9*(3), 398–410. https://www.researchgate.net/publication/262658887_The_relation_between_organizational_change_and_Information_Systems_deployment_a_study_in_auto_parts_industry

Skelton, A. (2018). *The Microsoft Dynamics Sure Step Methodology*. https://www.mercuriusit.com/the-microsoft-dynamics-sure-step-methodology/

Smith, G., & Sidky, A. (2009). *Becoming agile: In an imperfect world*. Manning Publications Co.

Solinski, A., & Petersen, K. (2016). Prioritizing agile benefits and limitations in relation to practice usage. *Software Quality Journal*, *24*(2), 447–482. https://doi.org/10.1007/s11219-014-9253-3

Sousa, J. C. A. de. (2018). *Estudo comparativo das metodologias ágeis e PMBOK* [Dissertação de Mestrado, Instituto Politécnico de Viseu]. https://repositorio.ipv.pt/handle/10400.19/4880

Stellman, A., & Greene, J. (2014). *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*. O'Reilly Media, Inc. https://books.google.pt/books?hl=en&lr=&id=XLxUBQAAQBAJ&oi=fnd&pg=PT33&dq=kanban+agile&ots=eVbo0Wjrfg&sig=LtSCPPK2XjnI5XbI5sAM5uB8G2M&redir_esc=y#v=onepage&q=kanban&f=false

Stoica, M., Mircea, M., & Ghilic-micu, B. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*, *17*(4), 64–76. https://doi.org/10.12948

Streule, T., Miserini, N., Bartlomé, O., Klippel, M., & de Soto, B. G. (2016). Implementation of Scrum in the Construction Industry. *Procedia Engineering*, *164*, 269–276. https://doi.org/10.1016/j.proeng.2016.11.619

Syspro. (2017). *IDEAL Implementation Methodology*. SYSPRO. https://www.syspro.com/dl/BR/SYSPRO-IDEAL-Implementation-ALL-BR.pdf

Tam, C., Moura, E. J. da C., Oliveira, T., & Varajão, J. (2020). The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*, *38*(3), 165–176. https://doi.org/10.1016/j.ijproman.2020.02.001

Tolvanen, J.-P. (1998). *Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence* (47th ed.). University Library of Jyväskylä. https://www.researchgate.net/publication/34052904_Incremental_Method_Engineering_with_Modeling_Tools_Theoretical_Principles_and_Empirical_Evidence

Uniyal, M., & Kalia, A. (2016). An empirical study of heavy weight and light weight software development process models. *International Advanced Research Journal in Science, Engineering and Technology*, *3*(11), 123–128. https://doi.org/10.17148/IARJSET.2016.31124

Varajão, J. (2016). Success Management as a PM Knowledge Area – Work-in-Progress. *Procedia Computer Science*, *100*, 1095–1102. https://doi.org/10.1016/j.procs.2016.09.256

Varajão, J. (2018). AgilIS – Agile Information Systems. DASI Project. Information Systems Department, University of Minho.

Varajão, J. (2019). Key Competences of Information Systems Project Managers | Request PDF. *International Journal of Information Technology Project Management*, *10*(3), 73–90. https://doi.org/10.4018/IJITPM.2019070105

Wells, H. (2012). How Effective Are Project Management Methodologies? An Explorative Evaluation of Their Benefits in Practice. *Project Management Journal*, *43*(6), 43–58. https://doi.org/10.1002/pmj.21302

Wijers, G. M. (1991). *Modelling Support in Information Systems Development*. Thesis Publishers. https://www.semanticscholar.org/paper/Modelling-support-in-information-systems-Wijers/8eec9868aa1ee4f4012582eef4a47d55f0af32c3

Xia, W., & Lee, G. (2004). Grasping the complexity of IS development projects. *Communications of the ACM*, *47*(5), 68–74. https://doi.org/10.1145/986213.986215

Xia, W., & Lee, G. (2005). The ability of information systems development project teams to respond to business and technology changes: A study of flexibility measures. *European Journal of Information Systems*, *14*(1), 75–92. https://doi.org/10.1057/palgrave.ejis.3000523

Xu, L. X. X., Wang Feng Yu, Lim, R., & Lua Eng Hock. (2010). A methodology for successful implementation of ERP in smaller companies. *Proceedings of 2010 IEEE International Conference on Service Operations and Logistics, and Informatics*, 380–385. https://doi.org/10.1109/SOLI.2010.5551547

Zamfir, R., & Paul, R. (2017). *Implementing SAP S4HANA, on-premise with SAP Activate* [Wiki]. SAP Community Wiki. https://wiki.scn.sap.com/wiki/display/ATopics/Implementing+SAP+S4HANA%2C+on-premise+with+SAP+Activate#ImplementingSAPS4HANA,on-premisewithSAPActivate-SAPActivate-Deploy

Zhou, Y. (2009). UniX Process, Merging Unified Process and Extreme Programming to Benefit Software Development Practice. *2009 First International Workshop on Education Technology and Computer Science*, *3*, 699–702. https://doi.org/10.1109/ETCS.2009.690

Apêndices

A.1 Forças e limitações do Modelo *Waterfall*

Forças do Modelo *Waterfall*		
Caraterísticas	**Autores**	**Expressões**
Processo bem documentado e estruturado	(McConnell, 1996)	"The waterfall model works especially well if you have a technically weak staff or an inexperienced staff because it provides the project with a structure that helps to minimize wasted effort." "The waterfall model is the most well-known lifecycle model (...)"
	(Matkovic e Tumbas, 2010)	"As a strictly defined model, it is characterised by standardised activities described in detail in all development phases;"
	(Balaji e Murugaiyan, 2012)	"Each phase appropriate documentation is followed for the eminence of the development."
	(Stoica et al., 2013)	"(...) the documentation and structure design are an advantage when new members join the team;" "(...) stages are implemented one at a time, in sequence;"
Fácil de perceber, utilizar e coordenar	(Balaji e Murugaiyan, 2012)	"(...) it's simple to employ."
	(Stoica et al., 2013)	"it is easy to understand and use;" "it is easy to coordinate due to the model rigidity (...)"
	(Sarker et al., 2015)	"Simple and easy to understand and use."
	(Eason, 2016)	"Waterfall methodology is known for having clear requirements, being easy to implement, use and manage."
Extensivamente testado e experimentado	(Roque, 1998)	"[Independently of the difference between the observed authors, the waterfall model, has characteristics that recommends it. Firstly this model has been extensively experimented and tested (...)]."
	(Avison e Fitzgerald, 2006)	"The SDLC has been well tried and tested (...)"
	(Balaji e Murugaiyan, 2012)	"The oldest of the SDLC's and the finest known."
Existência de maior controlo	(Roque, 1998)	"[Similarly, the division of the development of a system in stages allowed a better control over the development, fact that might, in theory, represent better results (...)]."
	(Avison e Fitzgerald, 2006)	"There are controls and these (...)"
Uso de documentação *standard*	(Avison e Fitzgerald, 2006)	"(...) and the use of documentation standards (...)"
	(Stoica et al., 2013)	"(...) the documentation and structure design are an advantage when new members join the team;"
Identificação de requisitos *a priori*	(Balaji e Murugaiyan, 2012)	"Prerequisite is clear before development commences."
	(Eason, 2016)	"Waterfall methodology is known for having clear requirements (...)"
Adequação a projetos de pequena dimensão	(Stoica et al., 2013)	"(...) it is recommended for small projects, with requirements clearly understood."
	(Sarker et al., 2015)	"Works well for smaller projects where requirements are very well understood and sufficient."
Adequação a projetos complexos e bem definidos	(McConnell, 1996)	"The waterfall model works well for projects that are well understood but complex."
Permite avaliação do progresso no final de cada etapa	(Roque, 1998)	"[It was verified, at the end of each stage, the opportunity to analysts and users to evaluate the obtained progress until that point]."
Garante a formação de todos os utilizadores	(Avison e Fitzgerald, 2006)	"The approach also ensures that users are trained to use the system."
Inclui uma etapa de teste	(Matkovic e Tumbas, 2010)	"It includes testing, i.e. verification of completed operations and obtained results at the closure of each development phase;"

Forças do Modelo *Waterfall*		
Caraterísticas	**Autores**	**Expressões**
Os indivíduos são facilmente substituídos	(Matkovic e Tumbas, 2010)	"Individual participants in the development process are comparatively easy to replace."
Cada etapa é realizada num período específico	(Balaji e Murugaiyan, 2012)	"Each phase is accomplished in specified period of time after that it moves to next phase."
Modelo linear	(Balaji e Murugaiyan, 2012)	"As it is a linear model, it's simples to employ."
Eficaz, eficiente e confiável em termos de engenharia	(Wells, 2012) *	"Waterfall is extremely engineering efficient, fitting requirements perfectly where you design once, develop once, and test once, and there is no need to write in a perfect environment where there is no change."

O símbolo * significa que se trata de um artigo com evidência empírica

Limitações do Modelo *Waterfall*		
Caraterísticas	**Autores**	**Expressões**
Inflexibilidade	(McConnell, 1996)	"Thus, the first major problem with the waterfall model is that it isn't flexible."
	(Roque, 1998)	"[Due to the formality degree, that demands specifications and documentation for each process that the information systems execute, changes are inhibited turning this process, a lot of times, inflexible to change]."
	(Avison e Fitzgerald, 2006)	"Inflexibility (due to the output-driven orientation of the design processes which makes changes in design costly);"
	(Matkovic e Tumbas, 2010)	"Inflexibility in the division of the development activities into separate phases and lack of back circuit i.e. feedback between stages;"
	(Balaji e Murugaiyan, 2012)	"Often, the client is not very precise of what he exactly wants from the software. Any changes that he reveals in between, may cause a lot of confusion."
	(Stoica et al., 2013)	"There is no flexibility in partitioning the project into stages."
	(Eason, 2016)	"(...) inability to change or update the project after the requirements have been defined."
Dificuldade em voltar atrás entre etapas	(McConnell, 1996)	"Some people have criticized the waterfall model for not allowing you to back up to correct your mistakes. That's not quite right. As Figure 7-1 suggests, backing up is allowed, but it's difficult."
	(Roque, 1998)	"[(...) waterfall model assumes that an activity must be concluded before the next one begins]."
	(Balaji e Murugaiyan, 2012)	"Sarcastically, the biggest shortcoming is one of its greatest benefits. You cannot go back a step; if the design phase has gone erroneous, things can get very problematical in the implementation phase"
	(Stoica et al., 2013)	"if testing detects some problems, it is very difficult to return to design stage;"
	(Sarker et al., 2015)	"Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage."
Documentação excessiva	(McConnell, 1996)	"(...) the waterfall model can prescribe an excessive amount of documentation."
	(Roque, 1998)	"[Due to the formality degree, that demands specifications and documentation for each process that the information systems execute (...)" "[The demand for voluminous formality, based on paper (...)]."
	(Avison e Fitzgerald, 2006)	"Problems with documentation (due to its computer rather than user orientation and the fact that it is rarely kept up-to-date);"
	(Eason, 2016)	"But there is also a high documentation (...)"
Demora na visualização de resultados/produtos funcional (apenas acontece no final do projeto)	(Roque, 1998)	"[The results observed in the post-implementation stage are time-consuming, that is, it will not occur until a greater number of steps have been completed. Most implementations of the waterfall life cycle rely on

Limitações do Modelo *Waterfall*		
Caraterísticas	**Autores**	**Expressões**
		sequential phases, which means that months or years can pass before the users can see any tangible evidence of progress]."
	(Matkovic e Tumbas, 2010)	"Lengthy development process, so that users have to be very patient and persistent, because the working versions of the software are unavailable before the end of the development activity, until when there is only a written specification of the future software's functionalities;"
	(Stoica et al., 2013)	"there are no prototypes until the life cycle is finished."
	(Sarker et al., 2015)	"No working software is produced until late during the life cycle."
A identificação de erros apenas ocorre no final do projeto e existe dificuldade em corrigi-los	(Roque, 1998)	"[The process of detecting errors in the classic cascading life cycle is reserved for the formal testing phase of the project. At this stage, pressure on the system's final development activities, such as detection of analysis and design errors, leads to situations where it is difficult to correct them in view of the cost associated with eliminating the errors.]." "[Use of the bottom-up implementation. The bottom-up implementation starts its work by testing the system modules, then subsystems, and finally the system. Thus, the most serious errors (system integrity) are found at the end and not at the beginning of the testing phase]."
	(Matkovic e Tumbas, 2010)	"Errors not removed in individual development phases, during product testing or verification, can have tremendous distortion impact on the development as a whole;"
	(Balaji e Murugaiyan, 2012)	"Small alterations or errors that come up in the completed software may cause a lot of problems."
	(Stoica et al., 2013)	"If testing detects some problems, it is very difficult to return to design stage;"
Custos elevados (tempo, dinheiro, etc.)	(Roque, 1998)	"Uma vez que considera que uma etapa deve ser iniciada após a conclusão das atividades da etapa anterior, é gasto uma quantidade razoável de tempo e esforços levantando informações, especificando e documentando-as a cada etapa para sua posterior utilização;" "(...) levam a situações onde toma-se difícil a correção destes tendo em vista o custo associado a eliminação dos erros;"
	(Matkovic e Tumbas, 2010)	"High development costs."
	(Stoica et al., 2013)	"New requirements added by the client lead to additional costs, because they cannot be solved in the current edition of the product;" "it is difficult to estimate the time and budget for each stage;"
Dificuldade da adaptação à incerteza	(Matkovic e Tumbas, 2010)	"Difficulty of adaptation to uncertainty which tends to exist at the start of the project, when it is very hard for the users to explicitly specify all their software requirements;"
	(Wells, 2012) *	"Traditional and waterfall methods are extremely efficient and extremely reliable and extremely unable to adapt."
	(Sarker et al., 2015)	"Not suitable for the projects where requirements are at a moderate to high risk of changing."
Dificuldade na identificação de requisitos *a priori*	(McConnell, 1996)	"The disadvantages of the pure waterfall model arise from the difficulty of fully* specifying requirements at the beginning of the project (...)"
	(Avison e Fitzgerald, 2006)	"Application backlog (due to the maintenance workload as attempts are made to change the system in order to reflect user needs);"
Desenvolvimento de sistemas obsoletos e não ambiciosos	(Roque, 1998)	"[(...) this fact can lead to a delay in the installation of the system, often making it obsolete when effectively put into operation]." "[If the user's requirements have been misinterpreted or misunderstood, or if the user changes the requirements during the subsequent design and implementation phase, the life cycle may not produce results for the actual problem determined]."

Limitações do Modelo *Waterfall*		
Caraterísticas	**Autores**	**Expressões**
	(Avison e Fitzgerald, 2006)	"Unambitious systems design (due to the emphasis on 'computerizing' the existing system);"
Altos riscos e níveis de incerteza	(Sarker et al., 2015)	"High amounts of risk and uncertainty."
	(Eason, 2016)	"Waterfall also has high risk and uncertainty."
Fraco envolvimento do cliente no ciclo de desenvolvimento	(McConnell, 1996)	"The waterfall model generates few visible signs of progress until the very end. That can create the perception of slow development—even if it isn't true. Customers like to have tangible assurances that their projects will be delivered on time."
Insatisfação do cliente	(Avison e Fitzgerald, 2006)	"User dissatisfaction (due to problems with the documentation and the inability for users to 'see' the system before it is operational);"
Incapacidade de realização de iterações	(Matkovic e Tumbas, 2010)	"Impossibility of iterations during the realisation of development, as these cause serious problems and confusion in model implementation;"
Processo de desenvolvimento longo	(Matkovic e Tumbas, 2010)	"Lengthy development process (...)"
Não adequado a projetos longos, complexos, orientados a objetos, e onde haja probabilidade de mudança de requisitos	(Sarker et al., 2015)	"Not suitable for the projects where requirements are at a moderate to high risk of changing."
	(Eason, 2016)	"It is not well suited for complex or object oriented projects and its better for shortterm projects."

O símbolo * significa que se trata de um artigo com evidência empírica

A.2 Forças e limitações do Modelo Incremental

Forças do Modelo Incremental		
Caraterísticas	**Autores**	**Expressões**
Cada entrega representa um produto funcional	(Stoica et al., 2013)	"Each stage delivers a working product, that meets some of the client requirements;"
	(Alshamrani e Bahattab, 2015)	"Each release delivers an operational product."
	(Sarker et al., 2015)	"Generates working software quickly and early during the software life cycle."
O feedback do cliente é obtido ao longo do processo de desenvolvimento (foco no cliente)	(Bhuvaneswari e Prabaharan, 2013)	"More focused on customer value than the linear approaches."
	(Stoica et al., 2013)	"(...) client feed-back is distributed throughout the entire development process;"
	(Sarker et al., 2015)	"Customer can respond to each built."
Priorização de tarefas	(Alshamrani e Bahattab, 2015)	"Customers get important functionality early and have an opportunity to respond to each build."
	(Sarker et al., 2015)	"(...) risky pieces are identified and handled during its iteration."
O trabalho é realizado através de pequenos incrementos (o que propicia a criação de valor desde cedo no projeto)	(Bhuvaneswari e Prabaharan, 2013)	"The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental) (...)"
	(Alshamrani e Bahattab, 2015)	"Risk is spread across smaller increments instead of concentrating in one large development."
Redução de custos nas entregas iniciais	(Stoica et al., 2013)	"Cuts down on initial delivery costs"
	(Sarker et al., 2015)	"Lowers initial delivery cost."
Maior facilidade na gestão de riscos	(Stoica et al., 2013)	"The risk is easier to manage because all risks are identified and managed during the iteration;"
	(Sarker et al., 2015)	"Easier to manage risk because risky pieces are identified and handled during its iteration."
Melhoria e aprendizagem contínua	(Alshamrani e Bahattab, 2015)	"Lessons learned at the end of each incremental delivery can result in positive revisions for the next increment."

Entrega mais rápida de um produto	(Alshamrani e Bahattab, 2015)	"Initial product delivery is faster."
Redução do risco de falha e mudança de requisitos	(Alshamrani e Bahattab, 2015)	"Reduces the risk of failure and changing the requirements."
Existência de protótipos funcionais	(Stoica et al., 2013)	"Prototypes are delivered to the client."
Maior facilidade de teste e *debug*	(Stoica et al., 2013)	"It is easy to test and debug during a small iteration;"
Melhor uso de recursos através da definição clara do incremento	(Bhuvaneswari e Prabaharan, 2013)	"Better use of scarce resources through proper increment definition."
Deteção de problemas mais cedo	(Bhuvaneswari e Prabaharan, 2013)	"Problems can be detected earlier."
Baixo custo de mudança de requisitos e âmbito	(Sarker et al., 2015)	"More flexible – less costly to change scope and requirements."
	(Stoica et al., 2013)	"It is more flexible – involves lower costs when purpose and requirements change;"

O símbolo * significa que se trata de um artigo com evidência empírica

Limitações do Modelo Incremental		
Caraterísticas	**Autores**	**Expressões**
Requer um bom planeamento e *design*	(Stoica et al., 2013)	"it requires a good planning and design;"
	(Alshamrani e Bahattab, 2015)	"Requires good planning and design."
	(Sarker et al., 2015)	"Needs good planning and design."
Requer a definição de todas as funcionalidades dos produtos *a priori*	(Stoica et al., 2013)	"requires a clear and complete definition of the entire system before it can be divided and incrementally built;"
	(Alshamrani e Bahatta, 2015)	"Requires early definition of a complete and fully functional system to allow for the definition of increments."
	(Sarker et al., 2015)	"Needs a clear and complete definition of the whole system before it can be broken down and built incrementally."
Custo mais elevado	(Stoica et al., 2013)	"Total cost is higher than the waterfall model."
	(Sarker et al., 2015)	"Total cost is higher than waterfall."
Não permite a realização de múltiplas iterações dentro do mesmo incremento	(Alshamrani e Bahatta, 2015)	"The model does not allow for iterations within each increment."
Pode tornar-se num ciclo de "*code and repair*"	(Stoica et al., 2013)	"Incremental approach may easily turn into "code and repair".
Requer muita documentação	(Bhuvaneswari e Prabaharan, 2013)	"Requires heavy documentation."
Segue um conjunto predefinido de processos	(Bhuvaneswari e Prabaharan, 2013)	"Follows a defined set of processes, defines increments based on function and feature dependencies."
Define os incrementos baseado na sua funcionalidade e dependências	(Bhuvaneswari e Prabaharan, 2013)	"Follows a defined set of processes, defines increments based on function and feature dependencies."
Requer maior envolvimento do cliente	(Bhuvaneswari e Prabaharan, 2013)	"It requires more customer involvement than the linear approaches."
Integração entre iterações pode ser um problema	(Bhuvaneswari e Prabaharan, 2013)	"Integration between iteration can be an issue if this is not considered during the development."

O símbolo * significa que se trata de um artigo com evidência empírica

A.3 Forças e limitações do Modelo em Espiral

Forças do Modelo em Espiral		
Caraterísticas	**Autores**	**Expressões**
Desenvolvimento de *software* funcional desde cedo no projeto e num curto período	(Matkovic e Tumbas, 2010)	"Production of functional software in a short time period;"
	(Bhuvaneswari e Prabaharan, 2013)	"Software is produced early in the software life cycle."

	(Alshamrani e Bahatta, 2015)	"Software is produced early in the software life cycle."
Valorização da análise de riscos	(Bhuvaneswari e Prabaharan, 2013)	"High amount of risk analysis."
	(Alshamrani e Bahatta, 2015)	"High amount of risk analysis."
	(Uniyal e Kalia, 2016) *	"It is clear from Table5 that 66% of the respondents were using spiral model for the development of risky projects (...)"
Flexibilidade na fase de engenharia (por exemplo, requisitos)	(Matkovic e Tumbas, 2010)	"Flexibility in the engineering phase (...);"
	(Alshamrani e Bahatta, 2015)	"Additional functionality can be added at a later date."
Bom para projetos grandes, críticos e de alto risco	(Bhuvaneswari e Prabaharan, 2013)	"Good for large and mission-critical projects."
	(Uniyal e Kalia, 2016) *	"Software professionals supported spiral model for critical projects where high risks are involved."
Identificação dos riscos desde o início do projeto	(Alshamrani e Bahatta, 2015)	"Provides early indication of insurmountable risks."
	Uniyal e Kalia (2016) *	"(...) because spiral model is the one that can identify risks early in the project."
Possibilidade de combinação de diferentes abordagens de desenvolvimento de *software*	(Matkovic e Tumbas, 2010)	"(...) and the possibility of combining various approaches to software development."
Possibilidade de analisar o risco a qualquer momento em qualquer nível de abstração	(Matkovic e Tumbas, 2010)	"Possibility of risk assessment at any time and abstraction level, thus providing for timely response to observed risks and a mechanism for their reduction by the application of prototype model;"
Abordagem sistemática do modelo waterfall, com possibilidade de iteração	(Matkovic e Tumbas, 2010)	"The model retains the systematic approach taken over from the waterfall model, with the possibility of iteration."
Forte controlo da documentação	(Alshamrani e Bahatta, 2015)	"Strong approval and documentation control."
Feedback constante, desde cedo, pelos utilizadores/cliente	(Alshamrani e Bahatta, 2015)	"Early and frequent feedback from users" "Concerned people of a project can early review each phase and each loop as well because of rapid prototyping tools."

O símbolo * significa que se trata de um artigo com evidência empírica

Limitações do Modelo em Espiral		
Caraterísticas	**Autores**	**Expressões**
Requer conhecimento específico para realização da análise de riscos	(Matkovic e Tumbas, 2010)	"(...) for risk analyses require specific expertise;"
	(Bhuvaneswari e Prabaharan, 2013)	"Risk analysis requires highly specific expertise."
	(Alshamrani e Bahatta, 2015)	"Risk assessment expertise is required."
Não é adequado para produtos ou projetos de pequena dimensão	(Matkovic e Tumbas, 2010)	"The model is comparatively capital-intensive for application in small-scale product development, for risk analyses require specific expertise;"
	(Bhuvaneswari e Prabaharan, 2013)	"It doesn't work well for smaller projects."
	(Alshamrani e Bahatta, 2015)	"Time spent for evaluating risks for small or low-risk projects may be too large."
Uma falha na análise de riscos tem um grande impacto no sucesso do projeto	(Matkovic e Tumbas, 2010)	"Great problems are created in situations when risks are not detected in due time or at all, thus producing multiplicator effect in its development."
	(Bhuvaneswari e Prabaharan, 2013)	"Project's success is highly dependent on the risk analysis phase."
Falta de *standards* para este modelo	(Matkovic e Tumbas, 2010)	"Absence of correlation with the existing standards, i.e. lack of standards for this software development model;"
Necessidade de maior uniformidade e consistência no desenvolvimento	(Matkovic e Tumbas, 2010)	"The model requires more uniformity and consistency in development;"

Limitações do Modelo em Espiral		
Caraterísticas	**Autores**	**Expressões**
Pode representar grandes riscos	(Bhuvaneswari e Prabaharan, 2013)	"Can be a costly model to use."
Tem grandes custos associados	(Alshamrani e Bahatta, 2015)	"Cost involved in this model is usually high."
A quantidade de documentação necessária nas etapas intermédias torna o processo mais complexo	(Alshamrani e Bahatta, 2015)	"Amount documentation required in intermediate stages makes management of a project very complex."

O símbolo * significa que se trata de um artigo com evidência empírica

A.4 Forças e limitações do *Unified Process*

Forças do *Unified Process*		
Caraterísticas	**Autores**	**Expressões**
Melhoria da qualidade do produto	(Rational Software, 1998)	"Better overall quality."
	(Kruchten, 2001)	"Improved software quality,"
	(Anwar, 2014)	"Improved management: high quality software that fulfills users' requirements is delivered regularly and on time." "The RUP greatly increases the quality of software produced."
Melhoria da comunicação entre a equipa, e com o cliente	(Kruchten, 2001)	"Improved team communication. Requirements management facilitates the involvement of users early in the process, helping to ensure that the application meets their needs. Well-managed requirements build a common understanding of the project needs and commitments among the stakeholders: users, customers, management, designers, and testers."
	(Osorio et al., 2011) *	"The iterative development approach also has an impact on the amount of feedback that the development team receives from the customer (...)"
	(Anwar, 2014)	"Regular feedback to stakeholders."
Entrega de um produto funcional com maior regularidade e mais cedo no projeto	(Zhou, 2009)	"There are cycles, each delivering a product release at the end."
	(Osorio et al., 2011) *	"RUP enables the development team to deliver value earlier and on a more frequent basis as a consequence of the iterative development approach."
	(Anwar, 2014)	"Regular feedback to stakeholders: stakeholders can quickly see portions of the system coming together;"
Melhor controlo sobre os projetos	(Kruchten, 2001)	"Better control of complex projects. This includes greater understanding of the intended system behavior as well as prevention of requirements creep."
	(Osorio et al., 2011) *	"The iterations provide project managers with a smaller control unit that enables a better scheduling and control of activities."
Identificação e mitigação dos riscos desde o início do projeto	(Rational Software, 1998)	"Risks are mitigated earlier."
	(Anwar, 2014)	"Improved risk management: working iteratively allows higher risks to be found and addressed early in the process."
A mudança é gerida apropriadamente	(Rational Software, 1998)	"Change is more manageable."
	(Anwar, 2014)	"You implement the actual requirements: change is unavoidable." "Changes to requirements in the current iteration are easy to deal with because an individual iteration has a small scope of requirements compared to a whole project. Changes to previous iterations are scheduled as new requirements for future iterations."
Maior satisfação do cliente	(Kruchten, 2001)	"Improved software quality and customer satisfaction."
Redução de custos e atrasos	(Kruchten, 2001)	"Reduced project costs and delays. Fixing errors in requirements is very expensive. With effective requirements management, you can decrease these errors early in the development, thereby cutting project costs and preventing delays."

Forças do *Unified Process*		
Caraterísticas	**Autores**	**Expressões**
Providencia todo o ciclo de desenvolvimento de *software*	(Zhou, 2009)	"UP provides a full software lifecycle solution."
Flexível e adaptável	(Zhou, 2009)	"In general, both UP and XP are requirement-driven, iterative and incremental. They are flexible and adaptive to fit a particular project."
Adequado a projetos de media e grande dimensão	(Zhou, 2009)	"(...) while UP is a heavy-weight methodology, working well with medium and large-size projects, not ready to face late requirement changes."
Requer planeamento no início de cada iteração	(Zhou, 2009)	"UP emphasizes the traceability and consistency of the large number of artifacts, including models, code, documentation (...)"
Melhora a previsão de resultados na fase de execução	(Osorio et al., 2011) *	"The activities performed during the inception and elaboration phases increase the predictability of the project execution (...)"
Melhor ajustamento aos requisitos do cliente	(Osorio et al., 2011) *	"(...) helps the team to adjust the software to better match to the customers expectations."
Melhores níveis de reutilização	(Rational Software, 1998)	"Higher level of reuse."
Melhoria e aprendizagem contínua	(Rational Software, 1998)	"The project team can learn along the way."
Definição da estrutura do projeto *a priori*	(Anwar, 2014)	"By developing a system skeleton during the elaboration phase, it greatly lessens the project technical risks."
Foco nas atividades mais importantes do projeto	(Anwar, 2014)	"Developers focus on what actually matters: when the RUP is implemented correctly, developers spend more time on actual software development;"

O símbolo * significa que se trata de um artigo com evidência empírica

Limitações do *Unified Process*		
Caraterísticas	**Autores**	**Expressões**
Existência de pequenas integrações ao longo da iteração, com uma integração final da iteração	(Zhou, 2009)	"On the side of UP, in the construction phase, there are series of builds and build integrations in an iteration, and one system integration at the end of the iteration. No integration frequency is specified."
	(Anwar, 2014)	"However, a minor disadvantage is there; need for integration of components later on. But this disadvantage is typically far outweighed by advantages."
Não preparação para a receção de alterações nos requisitos numa fase avançada do projeto	(Zhou, 2009)	"It is reasonable for the UP style to capture the user requirements as complete and accurate as possible at the beginning, either through an analyst going to meet the customer or the customer coming to tell the stories." "UP hopes to capture a high percentage of requirements as accurate as possible in the first two phases and assumes the cost of later change is high."
Orientado a casos de uso	(Zhou, 2009)	"UP is use case driven. Use case modeling is formal."
Elevado número de artefactos	(Zhou, 2009)	"The UP teams should maintain the large number of models throughout the software life cycle."
Fraco suporte para a gestão de projetos	(Osorio et al., 2011) *	"RUP was reported to offer inadequate support to project management."
Fraca integração dos stakeholders e do seu *feedback*	(Osorio et al., 2011) *	"(...) but RUP does not provide project managers with enough practical support to deal with the stakeholder integration."
Requer customização, tendo em conta a organização e a equipa do projeto, sendo um processo complexo, com grande custo, e que requer profissionais com experiência nesta atividade	(Osorio et al., 2011) *	"Although customizing RUP is absolutely necessary, it is an expensive and very complex process in itself that requires staff with a lot of experience and knowledge of RUP and there is no agreement about how to customize RUP."

O símbolo * significa que se trata de um artigo com evidência empírica

A.5 Forças e limitações dos modelos ágeis

Forças dos modelos ágeis		
Caraterísticas	**Autores**	**Expressões**
Habilidade de responder à mudança de requisitos	(Laanti, et al.,2011) *	"Requirements management/iterative planning." "However, planning was considered beneficial by both groups too. The negative group found beneficial the flexibility to take in new requirements any time. (...) The positive group saw agile planning and priorities as increasing predictability – and typically as simply better way to do planning, as is case also in this comment: ''Product plans are more realistic and thus is easier to plan other things related to products'"'. "The areas of requirements management and visibility and transparency were perceived as among the most beneficial (...)"
	(Balaji e Murugaiyan, 2012)	"The most important of the advantages of the agile model is the ability to respond to the changing requirements of the project."
	(McCormick, 2012)	"The most important advantages of this model are the ability to respond to the changing requirements of the project."
	(Kumar e Bhatia, 2012)	"Handling Change of Requirements."
	(Moniruzzaman e Hossain, 2013)	"Higher tolerant of change requirements: The main difference between heavyweight and agile methodologies is the acceptance of change."
	(Papatheocharous e Andreou, 2013) *	"enhanced the ability to manage changing requirements/priorities and contributed to software maintainability and extensibility."
	(Uniyal e Kalia, 2016) *	"The analysis and discussion leads to the conclusion that agile model is frequently used by software developers independent of size of the project, as it has the flexibility to change the requirements at any stage."
Entrega de um produto de melhor qualidade ao cliente	(Begel e Nagappan, 2007) *	"Not far behind was Improved Quality. The quality of the software is a strong concern of developers. The effects were manifested as fewer bugs, and a more stable set of features. (...) All aspects of software are improved, from design and architecture to performance of the products of each sprint."
	(McCormick, 2012)	"The culmination of this is that high quality software is delivered to the client in the shortest period of time and leaves the customer satisfied."
	(Kumar e Bhatia, 2012)	"Improvement in quality."
	(Moniruzzaman e Hossain, 2013)	"Improves software quality."
	(Uniyal e Kalia, 2016) *	"89% of the respondents were of the opinion that there was increase in the quality of software on using agile for small scale projects as compared to heavyweight."
	(Kamei et al., 2017) *	"Improved quality. This benefit was perceived mainly concerning the code and the project in general."
	(Laanti et al., 2011) *	"This study supports the findings of Dybå and Dingsøyr [31] about the benefits of (customer) collaboration and increased quality."
	(Papatheocharous e Andreou, 2013) *	"Furthermore, they improved software quality (...)"
Os testes ocorrem em cada iteração (Melhor deteção e resolução de problemas)	(Begel e Nagappan, 2007) *	"(...) and more reliance on test-driven."
	(Kumar e Bhatia, 2012)	"As testing is performed during each iteration (...)"
	(Mohammed et al., 2013)	"Another advantage of agile methodologies is the constant testing and integration since testing is done after each iteration which means that faults can be corrected regularly."
	(Kamei et al., 2017) *	"Facilitated problem-solving."
Maior satisfação das equipas	(Begel e Nagappan, 2007) *	"Rounding out the top ten benefits of Agile development were (...), increased morale (often tied to continuous integration with deliverables at the end of each sprint) (...)"
	(Papatheocharous e Andreou, 2013) *	"Furthermore, they improved software quality, project visibility/management and team morale."
	(Mohammed et al., 2013)	"Another positive point in agile methodologies is that developers are often more pleased (...)"
	(Solinski e Peterson, 2016) *	"Observation 2: The most significant internal benefit categories from adopting agile are knowledge and learning, employee satisfaction, social skill development,

Forças dos modelos ágeis		
Caraterísticas	**Autores**	**Expressões**
		and feedback and confidence. On the detailed level, agile makes people feel purposeful, improves knowledge transfer and learning between team members (...)"
Maior visibilidade e transparência do projeto	(Laanti, et al.,2011) *	"Visibility and transparency (key area 4) was found as another top benefit of agile development by both groups of respondents." "The areas of requirements management and visibility and transparency were perceived as among the most beneficial (...)"
	(Solinski e Peterson, 2016) *	"Agile leads to increased process control, transparency, and quality through continuous integration and small manageable tasks (Begel and Nagappan 2007; Petersen and Wohlin 2009)."
	(Mohammed et al., 2013)	"Another strength point is that progress can be viewed and measured after each iteration."
	(Kamei et al., 2017) *	"Improved project understanding. Scrum events as Sprint Planning and Review, and principle of Real Customer Involvement were important for this category."
Integração imediata de mudanças	(Begel e Nagappan, 2007) *	"Developers noted that short sprints combined with more emphasis on customer feedback led to better agility and efficiency in responding to changing requirements, internal processes, reorganizations or politics, and flushed out bad designs more quickly."
	(Laanti, et al.,2011) *	"Quality/testing/tools/continuous integration."
	(McCormick, 2012)	"The changes are integrated immediately, which saves trouble later."
Comunicação direta entre o cliente e a equipa de desenvolvimento	(Begel e Nagappan, 2007) *	"The top benefit was improved communication and coordination among team members." "Improved awareness of team members' activities was another benefit."
	(Balaji e Murugaiyan, 2012)	"There is no guesswork between the development team and the customer, as there is face to face communication and continuous inputs from the client."
	(McCormick, 2012)	"There is no guesswork between the development team and the customer, as there is face to face communication and continuous inputs from the client."
	(Kamei et al., 2017)	"Facilitated interaction and collaboration. A subset of benefits are related to this, as interaction and collaboration among the team, and a greater interaction between customers and team members."
	(Mohammed et al., 2013)	"Agile methodologies recommend having a full time customer working with development team. This means that any misunderstand of customers' requirements can be solved and answered immediately." "Another strength point is that the customer can evaluate the development after each iteration and suggest plans for future."
Entregas mais rápidas	(Begel e Nagappan, 2007) *	"The second most cited benefit was Quick Releases." "Developers create demo-able releases every few weeks instead of every few months or years."
	(Laanti, et al.,2011) *	"Frequent delivery/speed/responsiveness to change."
	(Moniruzzaman e Hossain, 2013)	"Agile development methodologies emphasize rapid delivery of software products to the clients."
No final de cada iteração há uma versão funcional do produto (Tempo reduzido de colocação no mercado)	(Moniruzzaman e Hossain, 2013)	"Agile development improvement in productivity, reduction development cost and reduction in time-to-market (Reifer, 2002)."
	(Papatheocharous e Andreou, 2013) *	"Agile methods helped in accelerating the time to market."
	(Mohammed et al., 2013)	"One of the strength points of agile methodologies is that its philosophy which based on the idea that projects are developed in short iterations. At the end of each iteration the user can see a working version of the software before moving ahead to the next iteration."
	(Kumar e Bhatia, 2012)	"Project delivery is divided into small functional releases or increments to manage risk and to get early feedback from customers and end users."
Maior flexibilidade	(Begel e Nagappan, 2007) *	"In third place is Flexibility of Design."
	(Kumar e Bhatia, 2012)	"Flexibility of design."
	(Mohammed et al., 2013)	"No doubt that the strength of agile methodologies is based on its flexibility."
Maior envolvimento do cliente e maior	(Begel e Nagappan, 2007) *	"Rounding out the top ten benefits of Agile development were better focus on customers (...)"

Forças dos modelos ágeis		
Caraterísticas	**Autores**	**Expressões**
obtenção do seu *feedback*	(Moniruzzaman e Hossain, 2013)	"Short design phase involves early feedback from clients."
	(Mohammed et al., 2013)	"In agile methodologies customers are playing an important role in the working team. Agile methodologies recommend having a full time customer working with development team. This means that any misunderstand of customers' requirements can be solved and answered immediately."
Entrega incremental, iterative e evolutiva	(Laanti, et al.,2011) *	"Requirements management/iterative planning." "However, [iterative] planning was considered beneficial by both groups too."
	(Kumar e Bhatia, 2012)	"Iterative and incremental delivery."
	(Moniruzzaman e Hossain, 2013)	"Agile software development methodologies are evolutionary and incremental models have become increasingly popular in software development industry."
Documentação na medida certa	(McCormick, 2012)	"The documents are to the point, which no leaves any space for ambiguity."
	(Moniruzzaman e Hossain, 2013)	"Agile approaches, emphasis more is on developing the application only, and not on documentation."
Melhoria do desempenho	(Kumar e Bhatia, 2012)	"Increased performance."
	(Papatheocharous e Andreou, 2013) *	"Increased productivity."
Priorização de requisitos	(Begel e Nagappan, 2007) *	"Rounding out the top ten benefits of Agile development were (...) better prioritization of development and focus on the product (...)"
	(Moniruzzaman e Hossain, 2013)	"In agile software development, requirements always provided by client and these requirement features are prioritized by client itself."
Redução de custos, e do tempo de entrega e desenvolvimento	(Moniruzzaman e Hossain, 2013)	"Reduce cost and time."
	(Uniyal e Kalia, 2016) *	"It is evident from Table3 that 54% software professionals were of the opinion that there decrease in the cost of the software on using agile for small scale projects."
	(Papatheocharous e Andreou, 2013) *	"Finally, they helped to increase productivity, reduce cost and risk, and in general simplified the development processes."
Facilidade de interação, colaboração, partilha e comunicação	(Uniyal e Kalia, 2016) *	"It was found that team members can communicate regularly throughout the development process while working on small projects."
	(Kamei et al., 2017) *	"Facilitated interaction and collaboration. A subset of bene□ts are related to this, as interaction and collaboration among the team, and a greater interaction between customers and team members."
	(Laanti et al., 2011) *	"People/communication/motivation/collaboration"
	(Mohammed et al., 2013)	"This means that there is always a continued channel for communication between team members which increase job satisfaction. Continued communications means identifying problems and mistakes earlier."
Maior produtividade, foco, eficiência e previsão	(Begel e Nagappan, 2007) *	"Rounding out the top ten benefits of Agile development were (...) improved productivity, (...)"
	(Laanti, et al.,2011) *	" Productivity/focus/efficiency/ predictability "
	(Kumar e Bathia, 2012)	"This increases team productivity and generates better performance in terms of good Return on Investment than the sum of all individual output."
Adequado para projetos pequenos	(Uniyal e Kalia, 2016) *	"89% of the respondents were of the opinion that there was increase in the quality of software on using agile for small scale projects as compared to heavyweight."
Satisfação do cliente	(McCormick, 2012)	"(...) and leaves the customer satisfied."
Desenvolvimento do projeto através de pequenas iterações	(Mohammed et al., 2013)	"One of the strength points of agile methodologies is that its philosophy which based on the idea that projects are developed in short iterations."
Baseado nas pessoas e na sua criatividade	(Mohammed et al., 2013)	"Also another strength point of agile methodologies is that it based on people and their creativity."
Equipas auto-organizadas	(Moniruzzaman e Hossain, 2013)	"Self organized team."
Design simples	(Moniruzzaman e Hossain, 2013)	"Design simplicity."
Aumento do valor, visibilidade e	(Moniruzzaman e Hossain, 2013)	"Increase Business value, visibility, adaptability and reduce cost."

Forças dos modelos ágeis		
Caraterísticas	**Autores**	**Expressões**
adaptabilidade do negócio		
Aumento da probabilidade de sucesso do projeto	(Moniruzzaman e Hossain, 2013)	"Success possibility increased."
Facilidade na monitorização e controlo do projeto	(Kamei et al., 2017) *	"Facilitated project monitoring and tracking. Many agile practices/events are useful to understand daily the health of the project."

O símbolo * significa que se trata de um artigo com evidência empírica

Limitações dos modelos ágeis		
Caraterísticas	**Autores**	**Expressões**
Difícil de aplicar em projetos de grande dimensão	(Begel e Nagappan, 2007) *	"The top concern of developers at Microsoft with Agile development is whether these methods scale to larger software teams." "Apprehensions concerning scaling to products with long release cycles or large legacy codebases were also mentioned."
	(Balaji e Murugaiyan, 2012)	"If the projects are smaller projects, then using agile model is certainly profitable, but if it is a large project, then it becomes difficult to judge the efforts and the time required for the project in the software development life cycle."
	(McCormick, 2012)	"If the projects are smaller projects, then using the agile model is certainly profitable, but if it is a large project, then it becomes difficult to judge the efforts and the time required for the project in the software development life cycle."
	(Kumar e Bhatia, 2012)	"Does not scale well to large projects, as numerous iterations are needed to complete the desired functionality."
	(Mohammed et al., 2013)	"One on the main weakness in agile methodologies is difficulty in coordinating between teams for large projects."
	(Solinski e Peterson, 2016) *	"It provides limited support for developing safety-critical systems and is not applicable to large teams." "The most significant limitations of utilizing agile practices in the development process are professional skill-specific demands and scalability." "Scalability issues need to be addressed, which were mainly related to distributed environments and applicability to large teams."
	(Uniyal e Kalia, 2016) *	"Whereas, project complexity was found one of the major problem when using agile for the large size projects." "But most of the respondents believed that there was decrease in quality of the software on using agile for medium and large-scale projects."
Requer profissionais seniores, com experiência ou especializados	(Balaji e Murugaiyan, 2012)	"Only senior developers are in a better position to take the decisions necessary for the agile type of development, which leaves hardly any place for newbie programmers, until it is combined with the senior resources."
	(McCormick, 2012)	"Only senior developers are in a better position to take the decisions necessary for the agile type of development (...)"
	(Mohammed et al., 2013)	"Beside that one of the key successes in agile methodologies based on having strong and high qualified developers having the creativity and skills to works in teams and to communicate with other teams or customers. This type of developers is not easy to find and demand a huge cost."
	(Solinski e Peterson, 2016) *	"The most significant limitations of utilizing agile practices in the development process are professional skill-specific demands and scalability."
	(Uniyal e Kalia, 2016) *	"Software professionals agreed that certain percentages of experienced people are required in an agile method."
	(Kamei et al., 2017) *	"Increased specialization of team members. Several agile principles emphasize that teams working on the same project should work together at the same place."
Falta de planeamento a longo prazo (previsibilidade)	(Begel e Nagappan, 2007) *	"In addition, because it is easy to move features to later sprints as work piles up, it is not easy to predict when a particular feature will go into the product."

Limitações dos modelos ágeis		
Caraterísticas	**Autores**	**Expressões**
	(Mohammed et al., 2013)	"Another point we can talk about that agile methodologies lack of long term planning."
	(Agrawal et al., 2016) *	"Lack of predictability."
Perceção errada sobre o conceito do *agile*	(Begel e Nagappan, 2007) *	"Many program managers were worried that upper-level management would ask for progress reports and productivity metrics that would be hard to gather in an Agile work environment. Management ignorance of Agile methodologies was also a worry." "The fourth concern was an apprehension about learning Agile development. Some developers wished they had formal training to do Agile, noting that there were few training options available to them."
	(Mohammed et al., 2013)	"Actually a very big mistake that many people assume that agile mainly based on flexibility and lack of flexibility. In reality this is not true, since agile methodologies has its own formal rules and polices."
	(Laanti et al., 2011) *	"The negative group interpreted agility as missing plans and planning as is also the case in this typical comment: ''Software is now created as fast as possible without any planning or design. Major decisions are made off the cuff in the hallway, based on whatever the project man- ager happens to think at the moment.''
Longos períodos podem ser atribuídos a pequenas funcionalidades	(Kumar e Bhatia, 2012)	"Too much time may be devoted to any single, small feature."
Requer muitos *meetings*	(Begel e Nagappan, 2007) *	"The second concern was about the Scrum. Scrum meetings involve all members of a team and often occur daily. Many respondents complained about the inefficiency of these meetings, especially when they were poorly run by a Scrum Master who was not disciplined and focused enough to run the meeting quickly."
	(Agrawal et al., 2016) *	"Table 4 and table 5 shows that performance is less affected by less predictability and lots of meetings limitation."
Falta de precisão do projeto antes do seu início	(Begel e Nagappan, 2007) *	"The rest of the top ten problems include culture clashes during Agile adoption, lack of a precise and overarching design before the project starts, the lack of a fixed and predictable schedule, and a perceived difficulty integrating developers and testers."
	(Agrawal et al., 2016) *	"Figure 1 shows that the biggest limitation of the agile methods is lack of upfront planning." "This work concludes that the major limitations of the agile methods are the lack of up front planning."
Interação com o cliente (o cliente pode não ter disponibilidade para estar com a equipa)	(Mohammed et al., 2013)	"Another point is customer interaction. Agile followers consider this point as a merit. But in fact it may become a weak point in some circumstances such as the user or customer might not find enough time to spend with developers, or if the key customer is one of the high level managers."
As equipas *agile* devem estar localizadas no mesmo espaço físico por longos períodos diários	(Mohammed et al., 2013)	"Another weakness point that agile methodologies recommend that the agile development team and end users are located in the same physical location for long periods daily which is not applicable under all situations."
Dificuldade na identificação de contributos individuais	(Mohammed et al., 2013)	"Also working in small teams guide to troubles in identifying individuals' contributions and how they can be pleased."
Dificuldade na identificação de como satisfazer os colaboradores	(Mohammed et al., 2013)	"Also working in small teams guide to troubles in identifying individuals' contributions and how they can be pleased."
Foco no processo levantamento de requisitos e no desenvolvimento de código, em vez que no design do produto	(Kumar e Bhatia, 2012)	"Main emphasis is on development rather than design and user. It basically focuses on processes for getting requirements and developing code and does not focus on product design."
Longos prazos para a realização de testes	(Kumar e Bhatia, 2012)	"High testing lead times and low test coverage."
Baixa cobertura dos testes	(Kumar e Bhatia, 2012)	"High testing lead times and low test coverage."
Requer boa coordenação e comunicação com o gestor de projetos	(Kumar e Bhatia, 2012)	"Many teams requiring high coordination and communication from project managers."

Limitações dos modelos ágeis		
Caraterísticas	**Autores**	**Expressões**
Pode representar grandes custos	(Kumar e Bhatia, 2012)	"On a large scale project, opportunity cost to employ agile methods may be too high for a foregone production on more profitable and lean projects."
Perda da perspetiva global do projeto	(Begel e Nagappan, 2007) *	"Losing sight of the big picture rounds out the top six concerns with Agile development."
Dificuldade na utilização de *user stories*	(Kamei et al., 2017) *	"Difficulty working with User Stories. A subset of limitations are related to this, as difficult to estimate, to split into tasks, to be used in complex projects, and to understand the user stories (...)"
Dificuldade em trabalhar em equipas grandes	(Kamei et al., 2017) *	"Difficulty working with large teams. Although one of the principles of Agile Methods is to work with small teams, this limitation is not new in literature, and was found in many interviews (...)"
Interferência do *product owner* com competências técnicas	(Kamei et al., 2017) *	"Interference of Product Owner with technical skills."
Dificuldade em aplicar/adaptar um modelo	(Kamei et al., 2017) *	"Difficulty applying/adapting the method/practice."
Restrições financeiras/de orçamento	(Agrawal et al., 2016) *	"Some other limitations are budget constraints (...)"
Falta de documentação	(Agrawal et al., 2016) *	"Figure 1 shows that the biggest limitation of the agile methods is lack of upfront planning followed by lack of sufficient documentation."
Não segue as etapas clássicas dos ciclos de vida de desenvolvimento de *software*	(Agrawal et al., 2016) *	"(...) unsupported SDLC steps." "Don't go with SDLC steps"
Requer formação	(Agrawal et al., 2016) *	"Requirement of training."
Não adequado para pequenas organizações	(Agrawal et al., 2016) *	"Not for small organizations."

O símbolo * significa que se trata de um artigo com evidência empírica

A.6 Forças e limitações do *Scrum*

Forças do *Scrum*		
Caraterísticas	**Autores**	**Expressões**
Melhores resultados são obtidos com maior rapidez	(Amlani, 2012)	"Fast moving developments can be quickly coded and tested using this method (...)"
	(Overhage e Schlauderer, 2012) *	"The interview results indicate that developers are more satisfied with the outcomes of Scrum projects"
	(Rubin, 2012)	"Fast results."
	(Obrutsky, 2016)	"Customer gets results faster."
Maior colaboração e comunicação	(Overhage e Schlauderer, 2012) *	"The results indicate that collaboration has increased likewise (H6)."
	(Law e Lárusdóttir, 2015) *	"More direct and effective communications enabled by daily stand-ups." "Enhanced team spirit."
	(Streule et al., 2016) *	"Mentioned benefits of Scrum were (...) better communication and collaboration (...)"
	(Azanha et al., 2017) *	"(...) more collaborative and productive aspects between developers and customers (...)" "In addition to these benefits indicated in project execution, we can highlight others, such as (...) the development team cooperation and motivation."
Entrega iterativa, frequente, continua, e mais rápida	(Amlani, 2012)	"(...) this is also iterative in nature."
	(Law e Lárusdóttir, 2015) *	"Value-driven, continuous delivery."
	(Azanha et al., 2017) *	"(...) faster delivery of product, better suited to the customer's reality and with the expected quality."

Forças do *Scrum*		
Caraterísticas	**Autores**	**Expressões**
		"Furthermore, frequent deliveries, provided by sprints, generate motivation to the project team, and constant communication." "(...) frequent deliveries and functionality increase, in short periods of time."
Obtenção de *feedback* do cliente continuamente	(Amlani, 2012)	"(...) It requires continuous feedback from the user."
	(Overhage e Schlauderer, 2012) *	"The interview results indicate that developers learn both from each other and from the customer."
	(Obrutsky, 2016)	"User feedback continues."
Maior satisfação dos *stakeholders*	(Rubin, 2012)	"Delighted customers."
	(Mahalakshmi e Sundararajan, 2013)	"Scrum provides customer satisfaction by optimizing turnaround time and responsiveness to requests."
	(Azanha et al., 2017) *	"And finally, but not least, it enhances customer satisfaction"
Melhoria da aprendizagem e da motivação da equipa	(Overhage e Schlauderer, 2012) *	"In comparison to traditional methodologies, there seem to be better learning effects in Scrum projects (H3)."
	(Streule et al., 2016) *	"Another important advantage of using Scrum was that the many meetings enabled the single member to see the point of view of other team members and starts to understand why something was done in a certain way. Thereby, team members improved their knowledge in a field where they were not experts, helping to support the concept of cross-functional teams."
	(Azanha et al., 2017) *	"Greater integration and commitment of the project team, who consequently feel more motivated: the team morale is boosted."
	(Law e Lárusdóttir, 2015) *	"Enabling reflection, knowledge sharing and transfer through retrospectives and stand-ups."
Maior transparência sobre o projeto	(Overhage e Schlauderer, 2012) *	"The study results furthermore suggest that the transparency of the development status in Scrum projects is better than in traditional projects (H5)."
	(Law e Lárusdóttir, 2015) *	"Visibility (or transparency): "the best thing both Scrum and Kanban is the visibility of what others are doing ... you know right way to see if something's wrong."(P4)."
	(Streule et al., 2016) *	"Mentioned benefits of Scrum were a higher transparency (...)"
Otimização de recursos (tempo, dinheiro, etc.)	(Amlani, 2012)	"Agile scrum helps the company in saving time and money."
	(Azanha et al., 2017) *	"(...) iterative approach, with incremental optimization of predictability and control risk."
Permite alcançar o sucesso do projeto onde a documentação e a identificação de requisitos é difícil de realizar e onde existe um grande nível de complexidade	(Amlani, 2012)	"Scrum methodology enables projects where the business requirements documentation is hard to measure to be successfully developed."
	(Rubin, 2012)	"Confidence to succeed in a complex world."
Aceitação e facilidade em realizar mudanças	(Amlani, 2012)	"Issues are identified well in advance through the daily meetings and hence can be resolved in speedily."
	(Mahalakshmi e Sundararajan, 2013)	"Accept and expect the changes." "Scrum is fast, quick and can adapt changes easily."
Flexibilidade e adaotabilidade à mudança	(Overhage e Schlauderer, 2012) *	"Even during later stages of the project, they may add or change requirements if necessary." "(...) The developers like the idea of taking over changes already during development and not having to work on change requests afterwards" (Scrum Coach 1)."
	(Obrutsky, 2016)	"Flexibility and adaptability to user changes."
Maior controlo sobre o projeto	(Mahalakshmi e Sundararajan, 2013)	"Be more in control of the project schedule."

Forças do *Scrum*		
Caraterísticas	**Autores**	**Expressões**
	(Azanha et al., 2017) *	"Another important proven benefit relates to the better project control, mainly its scope. This is due to the review meetings (sprint review) occurring at the end of each sprint (...)"
Realização de *sprints* de curta duração	(Mahalakshmi e Sundararajan, 2013)	"Freezes schedule – Short Sprint by short Sprint."
	(Law e Lárusdóttir, 2015) *	"The most frequently mentioned strength of Scrum was its effectiveness, because a finished feature would be delivered in each sprint. In addition, some mentioned that the short sprints of Scrum helped the team members to focus on a limited number of things at one time and that it was easy to plan and monitor individual as well as team progress."
Modelo sem grande controlo	(Amlani, 2012)	"(...) It is a lightly controlled method."
Facilidade em medir a produtividade individual	(Amlani, 2012)	"Daily meetings make it possible to measure individual productivity."
Rápida identificação e correção de problemas	(Amlani, 2012)	"Fast moving developments can be quickly coded and tested using this method, as a mistake can be easily rectified."
Funciona com qualquer tecnologia	(Amlani, 2012)	"Scrum can work with any technology/ programming language (...)"
Permite o levantamento de requisitos após entrega de um *deliverable*	(Obrutsky, 2016)	"It allows gather requirements after each deliverable."
Melhoria do retorno de investimento	(Rubin, 2012)	"Improved return on investment."
Redução de custos	(Rubin, 2012)	"Reduced costs."
Mais felicidade	(Rubin, 2012)	"More joy."
Aumento da qualidade	(Mahalakshmi e Sundararajan, 2013)	"Increase the quality."
Providencia melhores estimativas	(Mahalakshmi e Sundararajan, 2013)	"Provide better estimates while spending less time creating them."
Ideal para mudanças rápidas	(Mahalakshmi e Sundararajan, 2013)	"Scrum is ideal for rapidly changing, accumulating requirements."
Priorização de tarefas	(Mahalakshmi e Sundararajan, 2013)	"Estimates scope – Top feature, then next feature."
O trabalho a realizer numa *sprint* não é modificado	(Mahalakshmi e Sundararajan, 2013)	"Never changes the schedule, or Sprint."
Permite ajustar o âmbito do projeto	(Mahalakshmi e Sundararajan, 2013)	"Adjusts the scope if needed to meet release dates"
Estimar o trabalho a fazer é mais fácil de realizar	(Mahalakshmi e Sundararajan, 2013)	"Work estimates are much easier."
Redução do tempo de mercado	(Overhage e Schlauderer, 2012) *	"The interview results confirm that the developers perceive the time to market to be better in Scrum projects (H1)."
Melhor perceção dos requisitos	(Overhage e Schlauderer, 2012) *	"The results also confirm that the developers perceive the costumers' requirements to be better met in Scrum projects (H2)."
Melhor fluxo informacional	(Streule et al., 2016) *	"Mentioned benefits of Scrum were (...) better flow of information (...)."
Melhoria contínua	(Law e Lárusdóttir, 2015) *	"Continuous improvement through retrospectives."

O símbolo * significa que se trata de um artigo com evidência empírica

Limitações do *Scrum*		
Caraterísticas	**Autores**	**Expressões**
A alteração de um membro da equipa ou a sua fraca cooperação podem levar ao fracasso do projeto	(Amlani, 2012)	"If any of the team members leave during a development it can have a huge inverse effect on the project development."
	(Mahalakshmi e Sundararajan, 2013)	"If team members does not cooperate well, the project will face failure."
	(Obrutsky, 2016)	"If a member leaves the team, it decreases the team's productivity."
Não adequado para equipas de grande dimensão	(Amlani, 2012)	"It is good for small, fast moving projects as it works well only with small team."
	(Obrutsky, 2016)	"It is for small teams."
A gestão/controlo da qualidade do produtos é difícil se a equipa de testes não estiver disponível em cada *sprint*	(Amlani, 2012)	"Project quality management is hard to implement and measure if the test team are not able to conduct failure testing after each sprint."
	(Obrutsky, 2016)	"It is harder to implement quality controls because of the constant change."
Inexistência de uma data explícita para término de um projeto	(Amlani, 2012)	"Agile Scrum is one of the leading software development model because if there is not a definite end date, the project management stakeholders will be used to keep demanding new functionality to be delivered."
Dificuldade em estimar a duração e custo do projeto	(Amlani, 2012)	"If a task is not well defined, estimating project costs and time will not be accurate. In such a case, the task can be spread over several sprints."
Dependência do empenho dos diferentes membros de uma equipa	(Amlani, 2012)	"If the team members are not committed, the project will either never complete or fail."
Não adequado para projetos de grande dimensão	(Amlani, 2012)	"It is good for small, fast moving projects as it works well only with small team."
Necessidade de profissionais com experiência e com competências elevadas	(Amlani, 2012)	"This methodology needs experienced team members only. If the team consists of people who are new to this technology, the project cannot be completed in time."
Necessidade de confiança entre os membros de uma equipa	(Amlani, 2012)	"Scrum works well when the Scrum Master trusts the team they are managing. If they practice too strict control over the team members, it can be extremely frustrating for them, leading to demoralisation and the failure of the project."
Existência de pouca documentação	(Mahalakshmi e Sundararajan, 2013)	"Documentation is very less."
O trabalho em equipa é essencial	(Mahalakshmi e Sundararajan, 2013)	"Team work is highly essential."
Aumento da complexidade	(Overhage e Schlauderer, 2012) *	"On the other hand, the complexity seems to be increased by various factors. Especially, the teams had difficulties to get projects started."
Necessidade de maior disciplina	(Overhage e Schlauderer, 2012) *	"(...) the interview results show that the developers perceive the required discipline to be higher than in traditional projects."
Dificuldade na iniciação do uso do modelo	(Streule et al., 2016) *	"the disadvantages can be addressed to the low knowledge about Scrum at the beginning of the process."
Não identificação clara do líder do projeto	(Streule et al., 2016) *	"(...) the duties and responsibilities of each member were not clear or well defined."

O símbolo * significa que se trata de um artigo com evidência empírica

A.7 Análise comparativa das Forças identificadas

Modelos / Forças	Modelo Waterfall	Modelo em Espiral	Modelo incremental	*Unified Process*	Modelos ágeis	Scrum	Total de ocorrências
Software funcional desde cedo e num curto período/No final de cada iteração existe uma versão funcional do produto/Tempo reduzido de colocação do produto no mercado		X	X		X	X	4
O feedback do cliente é obtido ao longo do processo de desenvolvimento/foco no cliente/ Obtenção contínua de feedback do cliente			X		X	X	3
Priorização			X		X	X	3
O trabalho é realizado através de pequenos incrementos/iterações/sprints (o que propicia a criação de valor desde cedo no projeto; o trabalho a fazer pode ser modificado entre versões)			X		X	X	3
Entregas mais rápida/ Tempo reduzido de entrega			X		X	X	3
Melhor e mais rápida identificação, deteção e correção de problemas			X		X	X	3
Melhoria da qualidade do produto/ Entrega de um produto com maior qualidade/ Aumento da qualidade				X	X	X	3
Maior satisfação do cliente e dos *stakeholders*				X	X	X	3
Adequação a pequenos projetos	X				X		2
Flexibilidade na fase de engenharia (por exemplo, de requisitos)		X				X	2
Identificação e mitigação de riscos desde o início do projeto		X		X			2
Menor custo de alteração de requisitos e âmbito/ Permite ajustar o âmbito do projeto			X			X	2
Melhoria e aprendizagem contínua			X			X	2
Melhoria da comunicação da equipa e com o cliente/ Comunicação direta entre o cliente e a equipa de desenvolvimento				X	X		2
Maior e melhor controlo sobre os projetos				X		X	2
A mudança é aceite e gerida adequadamente				X		X	2
Habilidade de responder à mudança de requisitos					X	X	2
Equipas mais satisfeitas e com melhoria na motivação					X	X	2
Maior visibilidade e transparência do projeto					X	X	2
Entrega incremental, iterativa e evolutiva					X	X	2
Redução de custos					X	X	2
Facilidade na interação, colaboração, partilha e comunicação					X	X	2
Processo bem documentado e estruturado	X						1
Fácil de perceber, utilizar e coordenar	X						1
Extensivamente testado e experimentado	X						1
Existência de um maior controlo	X						1
Uso de documentação *standard*	X						1
Identificação de requisitos *a priori*	X						1
Adequação a projetos complexos e bem definidos	X						1
Permite avaliar o progresso no final de cada fase	X						1
Garante o treino de todos os utilizadores	X						1
Inclui uma etapa de teste	X						1
Os indivíduos podem ser facilmente substituídos	X						1
Cada etapa é complete num período específico	X						1
É um modelo linear	X						1
Extremamente eficiente, eficaz e confiável em termos de engenharia	X						1
Valorização da análise de riscos		X					1
Bom para projetos grandes, críticos e de alto risco		X					1
Possibilidade de combinar diferentes modelos de desenvolvimento de software		X					1
Possibilidade de analisar o risco a qualquer momento e nível de abstração		X					1
Redução de custos nas entregas iniciais			X				1
Maior facilidade na gestão de riscos			X				1
Redução do risco de falha e de mudança de requisitos			X				1
Existência de protótipos funcionais			X				1

Forças \ Modelos	Modelo Waterfall	Modelo em Espiral	Modelo incremental	*Unified Process*	Modelos ágeis	Scrum	Total de ocorrências
Maior facilidade em testar e fazer *debug*			X				1
Melhor uso dos recursos através de uma melhor definição do incremento			X				1
Entrega de produtos funcionais regularmente e desde cedo no projeto				X			1
Realização de testes em cada iteração					X		1
Integração imediata de mudanças					X		1
Maior flexibilidade					X		1
Documentação na medida certa					X		1
Melhoria da *performance*					X		1
Maior produtividade, foco, eficiência e capacidade de previsão					X		1
Baseado nas pessoas e na sua criatividade					X		1
Equipas auto-organizadas					X		1
Design simples					X		1
Aumento do valor, visibilidade e adaptabilidade do negócio					X		1
Aumento da probabilidade de sucesso do projeto					X		1
Melhores resultados obtidos mais rapidamente						X	1
Otimização de recursos						X	1
Permite alcançar o sucesso do projeto onde a documentação e a identificação de requisitos é difícil de realizar e onde existe grande complexidade envolvida						X	1
Modelo sem grande controlo						X	1
Facilidade em medir a produtividade individual						X	1
Funciona com qualquer tecnologia						X	1
Permite o levantamento de requisites depois da entrega de um *deliverable*						X	1
Melhoria do retorno de investimento						X	1
Maior felicidade						X	1
Providencia melhores estimativas						X	1
Ideal para a realização de mudanças rápidas						X	1
O trabalho a realizar numa iteração (sprint) não é modificado						X	1
Maior facilidade na estimativa do trabalho a realizar						X	1
Melhor perceção dos requisitos						X	1
Melhor fluxo informacional						X	1

A.8 Análise comparativa das limitações identificadas

Limitações \ Modelos	Modelo Waterfall	Modelo em Espiral	Modelo Incremental	*Unified Process*	Modelos ágeis	Scrum	Total de ocorrências
Custos elevados	X	X	X		X		4
Documentação excessiva	X	X	X				3
Requer profissionais séniores, com experiência ou especializados		X			X	X	3
Dificuldade na identificação dos requisitos/funcionalidades *a priori*	X		X				2
Requer maior envolvimento do cliente			X		X		2
Integração entre iterações pode ser um problema			X	X			2
Dificuldade na aplicação em projetos de grande dimensão					X	X	2
Dificuldade em trabalhar em grandes equipas					X	X	2
Falta de documentação					X	X	2
Não adequado a projetos pequenos		X					1
Não adequado a pequenas organizações					X		1
Inflexibilidade	X						1
Dificuldade em voltar atrás nas etapas	X						1

Modelos / Limitações	Modelo Waterfall	Modelo em Espiral	Modelo Incremental	*Unified Process*	Modelos ágeis	Scrum	**Total de ocorrências**
Demora na visualização de resultados/produtos funcional (apenas acontece no final do projeto)	X						1
A identificação de erros apenas ocorre no final do projeto e é difícil corrigi-los	X						1
Dificuldade na adaptação à incerteza	X						1
Desenvolvimento de sistemas obsoletos e não ambiciosos	X						1
Requer conhecimento específico para realização da análise de riscos		X					1
Uma falha na análise de riscos representa um grande impacto no sucesso do projeto		X					1
Falta de *standards*		X					1
Necessidade de maior uniformidade e consistência no desenvolvimento		X					1
Pode representar altos riscos		X					1
Requer um bom planeamento e *design*			X				1
Não permite a realização de múltiplas iterações sobre o mesmo incremento			X				1
Pode tornar-se num ciclo "*code and repair*"			X				1
Segue um conjunto predefinido de processos			X				1
Definição dos incrementos com base na sua função e dependências			X				1
Não preparação para a mudança de requisitos numa fase avançada do projeto				X			1
Orientado a casos de uso				X			1
Número elevado de artefactos				X			1
Fraco suporte para a gestão de projetos				X			1
Fraca integração dos stakeholders e do seu *feedback*				X			1
Requer customização, tendo em conta a organização e a equipa do projeto, o que é um processo complexa, e representa um grande custo e a necessidade de profissionais com experiência nessa atividade.				X			1
Falta de planeamento a longo prazo (previsibilidade)					X		1
Errada perceção do conceito agile (pressuposto de que estes modelos não possuem regras ou documentação)					X		1
Longos períodos podem ser atribuídos a uma funcionalidade pequena					X		1
Requer demasiados *meetings*					X		1
Falta de precisão no projeto antes deste começar					X		1
As equipas ágeis devem estar localizadas no mesmo espaço físico por longos períodos diariamente					X		1
Dificuldade na identificação de contributos individuais					X		1
Dificuldade na identificação de como satisfazer os colaboradores					X		1
Longos prazos para execução de testes					X		1
Baixa cobertura dos testes					X		1
Requer grande coordenação e comunicação com o gestor de projeto					X		1
Suporte limitado para o desenvolvimento de produtos críticos, complexos e de grande dimensão					X		1
Perda da perspetiva global do projeto					X		1
Dificuldade na utilização de *user stories*					X		1
Interferência do *product owner* com competências técnicas					X		1
Dificuldade na aplicação e adaptação do modelo					X		1
Restrições financeiras/do orçamento					X		1
Não segue as etapas clássicas dos ciclos de desenvolvimento de software					X	X	2
Requer formação					X		1
A remoção de um membro da equipa ou a fraca cooperação entre os seus membros pode levar ao fracasso do projeto						X	1
Dificuldade na gestão/controlo da qualidade se a equipa de teste não estiver disponível em cada *sprint*						X	1
Não existência de uma data específica para o término do projeto						X	1
Dificuldade na estimativa da duração e custo do projeto						X	1
Dependência do empenho dos diferentes membros de uma equipa						X	1
Necessidade de confiança entre os membros de uma equipa						X	1

Modelos / Limitações	Modelo Waterfall	Modelo em Espiral	Modelo Incremental	*Unified Process*	Modelos ágeis	Scrum	**Total de ocorrências**
O trabalho em equipa é essencial						X	1
Aumento da complexidade						X	1
Necessidade de maior disciplina						X	1
Dificuldade na iniciação do uso de um novo modelo						X	1
Não identificação clara do líder do projeto						X	1

A.9 Análise comparativa dos diferentes modelos de desenvolvimento de *software*

Com base na análise das forças e limitações, foi possível identificar algumas das principais caraterísticas dos modelos analisados, conforme representada na tabela apresentada neste apêndice e nos Apêndices A.7 e A.8.

Na tabela estão descritos os diferentes modelos, tendo em conta 27 elementos que emergiram da análise das forças e limitações:

1 Processo: fluxo de realização da atividade de um modelo;

2 Dimensão da organização: tamanho da organização a que o modelo mais se adequa (ou não);

3 Adequação a projetos: tipo de projetos a que se adequam;

4 Equipas do projeto: detalhes sobre a equipa do projeto;

5 Produtividade individual: medição da produtividade individual;

6 Colaboração e comunicação: níveis de comunicação e colaboração da equipa de um projeto;

7 Transparência: transparência sobre o projeto;

8 Priorização: priorização do trabalho a realizar;

9 Gestão do risco: identificação da forma de identificar e lidar com o risco;

10 Custos e incerteza: custos e níveis de incerteza associados ao modelo;

11 Controlo: nível de controlo do projeto e da equipa do projeto;

12 Aceitação da mudança: como é perspetivada a mudança num projeto;

13 Âmbito do projeto: como é gerido o âmbito do projeto;

14 Documentação: quantidade de documentação produzida num modelo;

15 Entrega: forma como é realizada a entrega do produto;

16 Qualidade: qualidade do *deliverable*;

17 Testes: comportamento da etapa de testes em cada modelo;

18 Estimativas: precisão das estimativas fornecidas;

19 Forma de trabalhar: formas de realizar o trabalho nos diferentes modelos;

20 Requisitos e mudança: comportamento face à identificação, levantamento e mudança de requisitos;

21 Complexidade: complexidade associada a cada modelo;

22 Conhecimento requerido: conhecimento necessário para utilização de cada modelo;

23 Envolvimento do cliente: nível de envolvimento do cliente ao longo do projeto;

24 Satisfação dos *stakeholders*: nível de satisfação dos diversos stakeholders do projeto;

25 Liderança: identificação da hierarquia de liderança no projeto;

26 Formação: forma como é realizada a formação em cada modelo;

27 Problemas e erros: forma como ocorre a identificação e resolução de problemas e erros no projeto.

	Modelo *Waterfall*	Modelo em Espiral	Modelo Incremental	*Unified Process*	Modelos ágeis	*Scrum*
Processo	**(+)** Processo estruturado e bem documentado; **(+)** Fácil de perceber, utilizar e coordenar; **(+)** Extensivamente testado e experimentado;	**(+)** Possibilidade de combinar diferentes abordagens de desenvolvimento de software; **(!)** Necessidade de maior uniformidade e consistência no desenvolvimento;	-	**(+)** Providencia todo o ciclo (clássico) de desenvolvimento de *software*;	**(=)** Não segue as etapas clássicas de desenvolvimento de *software*;	-
Dimensão da organização	-	-	-	**(+)** Pode ser customizado;	**(!)** Não adequado para pequenas organizações;	-
Adequação a projetos	**(!)** Adequando a projetos pequenos, complexos e bem definidos; **(!)** Não adequado a projetos longos, complexos, orientados a objetos e onde existe grande probabilidade de mudança de requisitos;	**(!)** Bom para projetos grandes, críticos e de alto-risco; **(!)** Não adequado a projetos pequenos;	-	**(-)** Fraco suporte para a gestão de projetos; **(!)** Adequado a projeto de média e grande dimensão;	**(!)** Adequado a projetos pequenos; **(!)** Difícil de aplicar em projetos grandes; **(!)** Suporte limitado para o desenvolvimento de projetos críticos, complexos e de grandes produtos; **(-)** Perda da visão global do projeto;	**(!)** Não adequado a projetos de grande dimensão; **(!)** Adequado para projetos onde a identificação de requisitos é difícil e tem elevada complexidade envolvida;

	Modelo *Waterfall*	Modelo em Espiral	Modelo Incremental	*Unified Process*	Modelos ágeis	*Scrum*
Equipas do projeto	-	-	-	**(+)** Melhoria da comunicação da equipa;	**(-)** As equipas ágeis devem estar localizadas no mesmo espaço físico por longos períodos diários; **(!)** Dificuldade em trabalhar em equipas grandes; **(+)** Equipas mais satisfeitas; **(+)** Equipas auto-organizadas;	**(!)** Não adequado para equipas grandes; **(=)** Requer confiança entre os membros de uma equipa; **(=)** O trabalho em equipa é essencial; **(=)** Requer maior disciplina; **(+)** Melhoria da motivação da equipa; **(+)** Mais felicidade;
Produtividade individual	-	-	-	-	**(-)** Dificuldade em identificar contributos individuais;	**(+)** Fácil de medir;
Colaboração e comunicação	-	-	-	-	**(=)** Requer boa coordenação e comunicação com o gestor de projeto; **(+)** Comunicação direta entre o cliente e a equipa de desenvolvimento; **(+)** Facilidade de interação, colaboração, partilha e comunicação;	**(+)** Melhoria da colaboração e comunicação;
Transparência	-	-	-	-	**(+)** Maior visibilidade e transparência do projeto;	**(+)** Maior transparência sobre o projeto;
Priorização	**(-)** Não	–	**(+)** Sim	**(+)** Sim	**(+)** Sim	**(+)** Sim
Gestão do risco	-	**(+)** Identificação de riscos desde o início do projeto; **(+)** Valorização da análise de riscos; **(+)** Possibilidade de analisar o risco em qualquer momento e nível de abstração;	**(+)** Maior facilidade na gestão de riscos;	**(+)** Os riscos são identificados e mitigados desde o início do projeto;	-	-

	Modelo *Waterfall*	Modelo em Espiral	Modelo Incremental	*Unified Process*	Modelos ágeis	*Scrum*
Custos e incerteza	**(-)** Custos elevados; **(-)** Riscos elevados **(-)** Elevado nível de incerteza;	**(-)** Pode representar elevados riscos; **(-)** Tem custos elevados associados;	**(-)** Custos elevados; **(+)** Redução de custos nas entregas iniciais;	**(+)** Redução de custos e atrasos;	**(-)** Pode representar grandes custos; **(+)** Redução de custos;	**(+)** Redução de custos;
Controlo	**(+)** Existência de um maior controlo;	-	-	**(+)** Melhor controlo dos projetos;	**(+)** Facilidade de monitorizar e controlar o projeto;	**(+)** Melhor controlo sobre o projeto; **(+)** Modelo sem grande controlo;
Aceitação da mudança	**(-)** Inflexibilidade;	-	–	**(+)** A mudança é gerida apropriadamente; **(+)** Flexível e adaptável;	**(+)** Integração imediata de mudanças;	**(+)** Aceitação e facilidade em realizar mudanças; **(+)** Flexibilidade e adaptabilidade à mudança; **(+)** Ideal para mudanças rápidas;
Âmbito do projeto	-	**(+)** Permite a mudança de âmbito;	-	-	-	**(+)** Permite o ajustamento do âmbito;
Documentação	**(-)** Documentação excessiva; **(+)** Uso de documentação standard;	**(-)** A quantidade de documentação necessária nas fases intermédias tornam o processo mais complexo; **(-)** Forte controlo da documentação;	**(-)** Requer muita documentação;	**(-)** Elevado número de artefactos (documentação);	**(+)** Documentação na medida certa;	**(+)** Existência de pouca documentação;
Entrega	**(-)** Demora na visualização de resultados (apenas no final do projeto); **(-)** Pode levar ao desenvolvimento de sistemas obsoletos e não ambiciosos;	**(+)** Entrega desde cedo no projeto; **(+)** No final de cada iteração; **(+)** *Software* funcional desde cedo no projeto e num curto período de tempo;	**(+)** Entrega mais rápida de um produto; **(+)** Desde cedo no projeto; **(+)** No final de cada iteração; **(+)** Pequenos incrementos; **(+)** Cada iteração representa um produto funcional;	**(+)** Entrega de um produto funcional com maior regularidade e desde cedo no projeto;	**(+)** Desde cedo no projeto; **(+)** No final de cada iteração; **(=)** Incremental; **(=)** Iterativa e evolutiva; **(=)** Pequenos incrementos; **(+)** Entregas mais rápidas; **(+)** Reduzido tempo de colocação no Mercado;	**(=)** Iterativa; **(+)** Frequente; **(+)** Contínua; **(+)** Mais rápida; **(+)** Reduzido tempo de colocação no mercado;
Qualidade		-	-	**(+)** Melhoria da qualidade do produto;	**(+)** Entrega de um produto de maior qualidade;	**(+)** Maior qualidade;

	Modelo *Waterfall*	Modelo em Espiral	Modelo Incremental	*Unified Process*	Modelos ágeis	*Scrum*
Testes	**(+)** Inclui uma fase de teste;	-	**(+)** Maior facilidade de testar e fazer *debug*;	-	**(-)** Longos prazos para execução de testes; **(+)** Ampla cobertura dos testes; **(+)** Os testes ocorrem em cada iteração;	-
Estimativas	-	-	-	-	-	**(+)** Providencia melhores estimativas; **(+)** A estimativa do trabalho a realizar é mais fácil;
Forma de trabalhar	**(=)** Sequencial; **(+)** Permite avaliação do progresso em cada fase; **(+)** Cada fase é complete num período específico; **(-)** É difícil voltar atrás nas fases; **(-)** Identificação de requisitos *a priori*; **(-)** Impossibilidade de realizar iterações;	**(=)** Abordagem sistemática do modelo waterfall, com possibilidade de iteração;	**(-)** Requer a definição de todas as funcionalidades do produto *a priori*; **(=)** Não permite a realização de múltiplas iterações dentro do mesmo incremento; **(=)** Segue um conjunto predefinido de processos; **(-)** Integração entre iterações pode ser um problema; **(+)** Existência de protótipos funcionais;	**(=)** Pequenas integrações ao longo da iteração, com uma integração no final na iteração; **(=)** Requer planeamento no início de cada iteração; **(+)** Definição da estrutura do projeto *a priori*;	**(=)** Desenvolvimento do projeto através de pequenas iterações;	**(=)** *Sprints* de curta duração;
Requisitos e mudança	**(-)** Identificação de requisitos *a priori*;	**(+)** Flexibilidade na fase de engenharia (por exemplo, de requisitos); **(+)** Baixo custo de mudança de requisitos ou âmbito;	-	**(+)** Melhor ajuste aos requisitos do cliente;	**(+)** Habilidade de responder à mudança de requisitos;	**(+)** Permite o levantamento de requisitos após a entrega de um *deliverable*; **(+)** Melhor perceção dos requisitos;
Complexidade	-	-	-	-	-	**(-)** Maior complexidade;
Conhecimento requerido	-	**(-)** Requer conhecimento específico para a realização da análise de riscos;	-	**(-)** Necessidade de profissionais com experiência (para customização);	**(-)** Requer profissionais seniores, com experiência ou especializados;	**(-)** Necessidade de profissionais com experiência ou com elevadas competências;

	Modelo *Waterfall*	Modelo em Espiral	Modelo Incremental	*Unified Process*	Modelos ágeis	*Scrum*
Envolvimento do cliente	**(-)** Baixo envolvimento do cliente no ciclo de desenvolvimento do projeto;	**(+)** Constante *feedback*, desde cedo, do cliente e/ou utilizadores;	**(+)** Requer maior envolvimento do cliente; **(+)** *Feedback* do cliente obtido ao longo do processo de desenvolvimento do projeto;	**(-)** Fraca integração dos *stakeholders* e do seu *feedback*; **(+)** Melhoria da comunicação com o cliente;	**(+)** Maior envolvimento e *feedback* do cliente;	**(+)** *Feedback* contínuo;
Satisfação dos *stakeholders*	**(-)** Insatisfação do cliente;	-	-	**(+)** Maior satisfação do cliente;	**(-)** Dificuldade em identificar como satisfazer os colaboradores; **(+)** Satisfação do cliente;	**(+)** Maior satisfação dos *stakeholders*;
Liderança	-	-	-	-	-	**(-)** Não identificação clara da liderança do projeto;
Formação	**(+)** Assegura a formação de todos os utilizadores;	-	-	-	-	-
Problemas e erros	**(-)** Apenas acontece no final do projeto e existe dificuldade em corrigi-los;	-	**(+)** Deteção de problemas cedo;	-	**(+)** Melhor deteção e resolução de problemas;	**(+)** Rápida identificação e correção de problemas;