ISCTE Business School
Instituto Universitário de Lisboa

# APLICAÇÃO DO MODELO *SCRUM* AO PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE* DA CGD

Paulo Jorge Espírito Santo Heitor

Projeto submetido como requisito parcial para obtenção do grau de Mestre em Gestão

Orientador:
Prof. Doutor Henrique Manuel Caetano Duarte, Prof. Auxiliar, ISCTE Business School, Departamento de Recursos Humanos e Comportamento Organizacional

Setembro 2015

**AGRADECIMENTOS**

Ao Professor Henrique Duarte agradeço a disponibilidade e orientação.

Agradeço ao Mário e ao Renato o trabalho conjunto que temos realizado no terreno ao longo destes vários anos, com muitas discussões saudáveis que me ajudam a crescer diariamente enquanto profissional.

Ao Eugénio agradeço a oportunidade e confiança demonstradas.

Aos meus pais, Fernanda e José, à Sofia e ao meu pequeno Tomás, agradeço o apoio dado ao longo desta etapa.

**RESUMO**

O desenvolvimento de *software* é uma atividade complexa, caracterizada por um elevado grau de variabilidade ao longo da sua execução. Os processos de desenvolvimento de *software* – conjuntos de atividades relacionadas que conduzem à realização de um produto de *software* - encontram-se em permanente evolução, influenciados quer pela constante alteração tecnológica, quer pela alteração das necessidades dos utilizadores dos produtos de *software*. Como parte desta evolução, as organizações têm vindo a substituir os seus processos de desenvolvimento de *software,* baseados em modelos Tradicionais, por processos baseados em modelos *Agile*. A alteração aos processos de desenvolvimento de *software* acarreta desafios enormes para as organizações, em particular para as pessoas.

O presente trabalho, realizado no âmbito do banco Caixa Geral de Depósitos (CGD), aborda os fatores que afetam diretamente as pessoas, na aplicação do modelo *Scrum* – modelo de processo de desenvolvimento de *software Agile* - ao processo de desenvolvimento de *software,* tendo como objeto a definição de um conjunto de ações influenciadoras do sucesso da sua adoção.

Foi escolhido o formato projeto empresa, para responder às seguintes questões de investigação:

1. Qual deverá ser o desenho do futuro processo de desenvolvimento de *software*?
2. Que medidas deverão ser realizadas para que a adoção do *Scrum* ocorra com sucesso na organização?

Foi realizado um diagnóstico situacional, o qual permitiu identificar o atual processo de desenvolvimento de *software*, a experiência que a CGD já tem relativamente à utilização do modelo *Scrum* e o estado em que se encontram os fatores identificados como influenciadores do sucesso da adoção do modelo *Scrum* ao processo de desenvolvimento de *software*.

A solução a adotar foca-se em 2 vertentes: (i) Futuro Processo de Desenvolvimento de *Software*; (ii) Medidas que influenciam o sucesso da adoção do *Scrum.* Adicionalmente foi caracterizado o projeto de implementação da solução apresentada.

Os resultados permitiram concluir que a solução encontrada, em resposta à questão da investigação, influencia positivamente os fatores de sucesso que afetam diretamente as pessoas na transição dos modelos *Waterfall* para modelos *Agile,* permitindo a adoção de um novo processo de desenvolvimento de *software* com benefícios evidentes para a organização.

Palavras-chave: Adoção do *Scrum*, *Agile,* Fatores de Sucesso, Desenvolvimento de *Software*.

**ABSTRACT**

Software development is a complex activity, characterized by a high degree of variability throughout its execution. Software development processes are constantly evolving, influenced either by constant technological change, either by the changing needs of software products users. As part of this evolution, organizations have come to replace their software development processes based on Traditional models for processes based on Agile models. Changing the software development processes entails enormous challenges for organizations, particularly for people.

This work, carried out under the bank Caixa Geral de Depósitos (CGD), addresses the factors that directly affect people, in the application of Scrum model to the software development process, having as its object the definition of a set of actions that influence success of its adoption.

The company project format was chosen to answer the following research questions:

1. What should be the design of the future process of software development?

2. What will be done for the adoption of Scrum successfully occurs in the organization?

A situational diagnosis was made, which allowed to identify the current software development process, the experience that CGD already has using Scrum model and the state in which they are the factors identified that influence the successful adoption of Scrum model to the software development process.

The solution to adopt focuses in 2 parts: (i) Future Software Development Process; (ii) measures that influence the successful adoption of Scrum. Additionally it was characterized the implementation project of the solution presented.

The results showed that the solution in response to the question of research, positively influences the success factors that directly affect the people in the transition from Waterfall models for Agile models, allowing the adoption of a new software development process with clear benefits for the organization.

Keywords: Scrum Adoption, Agile, Success Factors, Software Development.

## ÍNDICE GERAL

1 SUMÁRIO EXECUTIVO ..... 7
2 DEFINIÇÃO DO CONTEXTO DO PROBLEMA ..... 10
3 REVISÃO DE LITERATURA ..... 12
3.1 Processos de desenvolvimento de *software* ..... 12
3.1.1 Modelos Tradicionais ..... 13
3.1.2 Modelos Agile ..... 20
*3.2* Modelo *Scrum* ..... 22
3.2.1 Artefactos ..... 23
3.2.2 Eventos ..... 23
3.2.3 Papéis e Responsabilidades ..... 25
3.3 Coexistência entre os modelos *Waterfall* e *Scrum* ..... 27
3.4 Transição dos modelos Tradicionais para os modelos *Agile* ..... 29
3.4.1 Principais diferenças entre os modelos Tradicionais e Agile ..... 29
3.4.2 Formas de transição ..... 31
3.4.3 Fatores de sucesso relativos às pessoas ..... 33
4 QUESTÕES RESULTANTES DA REVISÃO DE LITERATURA ..... 41
5 MÉTODOS E TÉCNICAS DE RECOLHA E ANÁLISE DE DADOS ..... 43
6 A CAIXA GERAL DE DEPÓSITOS ..... 45
7 ANÁLISE DE INFORMAÇÃO E CONCLUSÕES ..... 47
7.1 Diagnóstico situacional ..... 47
7.1.1 Atual processo de desenvolvimento de software ..... 47
7.1.2 A utilização do Scrum no contexto atual ..... 52
7.1.3 Avaliação aos fatores que influenciam o sucesso da adoção do Scrum ..... 53
7.2 Solução a adotar ..... 56
7.2.1 Futuro processo de desenvolvimento ..... 56
7.2.2 Medidas que influenciam o sucesso da adoção do Scrum ..... 64

8 IMPLEMENTAÇÃO ..... 71
8.1 Organização do projeto ..... 71
8.2 Cronograma ..... 72
8.3 Custos de Implementação ..... 74
8.4 Avaliação da implementação ..... 76
8.5 Valor acrescentado ..... 77
9 CONCLUSÕES ..... 79
9.1 Limitações ..... 79
9.2 Futuro processo de desenvolvimento de *software* ..... 80
9.3 Medidas influenciadoras do sucesso da adoção do *Scrum* ..... 81
9.4 Contributo para a Caixa Geral de Depósitos ..... 83
10 BIBLIOGRAFIA ..... 84

## ÍNDICE DE FIGURAS

Figura 1 - Modelo *waterfall* ..... 14
Figura 2 - Desenvolvimento incremental ..... 16
Figura 3 – Engenharia de *Software* orientada à reutilização ..... 18
Figura 4 – Modelo dinâmico do *Scrum* ..... 27
Figura 5 – Fases do atual processo de desenvolvimento de *software* da CGD ..... 47
Figura 6 – Fases do novo processo de desenvolvimento da CGD ..... 58
Figura 7 - Atividades da fase de desenvolvimento do novo processo de *software* ..... 59
Figura 8 - Cronograma do projeto ..... 74

## ÍNDICE DE TABELAS

Tabela 1 - Atividades fundamentais para a engenharia de *software* ..... 12
Tabela 2 - Fases do modelo de processo de *software Waterfall* ..... 14
Tabela 3 - Fases do modelo de engenharia de *software* orientada à reutilização ..... 18
Tabela 4 - Descrição dos principais modelos *Agile* ..... 20
Tabela 5 - Artefactos do *Scrum* ..... 23
Tabela 6 - Eventos do *Scrum* ..... 23
Tabela 7 - Papéis e responsabilidades do *Scrum* ..... 25
Tabela 8 - Diferenças entre os modelos tradicional e o *Agile* ..... 30
Tabela 9 - Fatores de sucesso na adoção de modelos *Agile* ..... 34
Tabela 10 - Principais razões da resistência à mudança ..... 36
Tabela 11 - Ações que influenciam a auto-organização das equipas ..... 39
Tabela 12 - Fases do processo de desenvolvimento de *software* da CGD ..... 47
Tabela 13 - Papéis do processo de Desenvolvimento de *software* da CGD ..... 50
Tabela 14 - Atual processo CGD segundo variáveis de *Conboy* ..... 51
Tabela 15 - Descrição das atividades da fase de desenvolvimento ..... 59
Tabela 16 - Revisão dos Papéis e responsabilidades ..... 62
Tabela 17 - Recomendações de maximização fatores sucesso da adoção *Scrum* ..... 65
Tabela 18 - Custos de Implementação do novo processo de *software* ..... 75
Tabela 19 - *Milestones* do projeto de implementação ..... 76

# 1 SUMÁRIO EXECUTIVO

Este trabalho de investigação, com o formato projeto empresa, é realizado no âmbito do banco Caixa Geral de Depósitos (CGD). O projeto aborda a temática da transição de processos de desenvolvimento de *software* baseados em modelos Tradicionais, para processos baseados em modelos *Agile*, tendo como objeto a aplicação do modelo *Scrum* ao processo de desenvolvimento de *software* na CGD.

Atualmente, com um processo de desenvolvimento de *software* baseado no modelo Tradicional *Waterfall*, a CGD enfrenta os habituais problemas deste tipo de processos, nomeadamente: (i) Elevado esforço na realização e aprovação da documentação; (ii) Elevado custo de mudança, no que respeita a alterações ao plano traçado; (iii) Entrega de valor para o cliente apenas no final do processo; (iv) Problemas são empurrados para as fases mais tardias do processo; (v) Pouca transparência do processo e fraco envolvimento do cliente e (vi) Integração das peças de *software* e respetivos testes somente no final.

No ano de 2010 foi iniciada a utilização do modelo *Scrum*, por uma das suas equipas de desenvolvimento, a título experimental. Passados cinco anos esta equipa continua a utilizar o modelo *Scrum* evidenciando os seguintes benefícios: (i) Maior foco no objetivo; (ii) Entregas de valor para o negócio mais rápidas; (iii) Elevada motivação da equipa; (iv) Satisfação dos *Stakeholders*. Baseada nesta experiência bem-sucedida, a CGD pretende aplicar o modelo *Scrum* ao seu processo de desenvolvimento de *software*.

Perante este cenário a CGD equaciona a adoção do modelo *Scrum* ao seu processo de desenvolvimento de *software*.

Tendo por base este contexto, são colocadas 2 questões de investigação a responder:

1. Qual deverá ser o desenho do futuro processo de desenvolvimento de *software*?
2. Que medidas deverão ser realizadas para que a adoção do *Scrum* ocorra com sucesso na organização?

A revisão da literatura, realizada no âmbito deste trabalho de investigação, encontra-se focada nos modelos de processo de desenvolvimento *software*, na coexistência entre os mesmos e na transição de processos Tradicionais para processos *Agile*. Neste âmbito, são pesquisados os seguintes temas: (i) Processos de desenvolvimento *software*; (ii) Modelos de processos de desenvolvimento de *software* Tradicionais; (iii) Modelos de processos de desenvolvimento de *software Agile*; (iv) Coexistência entre os modelos *Waterfall* e *Scrum* e (v) Transição dos modelos Tradicionais para os modelos *Agile*.

A resposta às questões da investigação encontra-se estruturada da seguinte forma:

1. É analisado o atual processo de desenvolvimento de *software*, a experiência da organização com o modelo *Scrum* e estado em que se encontram os fatores identificados como influenciadores do sucesso da adoção do *Scrum*;
2. É definido o novo processo de desenvolvimento de *software*;
3. São definidas as medidas a implementar para garantir o sucesso da transição.

O diagnóstico situacional permitiu concluir o seguinte:

1. O atual processo de desenvolvimento de *software* é baseado no modelo *Waterfall*.
2. Os colaboradores percecionam as desvantagens do processo existente, nomeadamente: (i) Especificação detalhada no início do processo; (ii) Longos ciclos de desenvolvimento; (iii) Muita documentação; (iv) Elevado custo de mudança; (v) Entrega tardia de valor para o cliente; (vi) Fraco envolvimento do cliente; (vii) Inexistência de uma visão global na constituição de requisitos; (viii) Falta de autoridade e capacidade crítica das equipas de desenvolvimento; (ix) Testes somente no final do processo.
3. Existem pessoas com experiência na utilização do modelo *Scrum* desde o ano de 2010, as quais referem os benefícios da aplicação do modelo.
4. A formação no modelo *Scrum* já faz parte do plano de formação (opcional) dos colaboradores, existindo uma grande aceitação, predisposição e motivação dos formandos para a aplicação do modelo nas suas equipas.
5. Os planos de formação dos colaboradores, por norma, não abrangem os colaboradores externos, os quais representam a maioria dos elementos das equipas de desenvolvimento.
6. Embora os fatores de sucesso que influenciam positivamente a adoção do *Scrum,* necessitem de ser trabalhados através das medidas a aplicar na fase de implementação do projeto, existem pelo menos três fatores para os quais a CGD já iniciou um trabalho prévio: (i) Motivação; (ii) Compreensão dos valores e princípios *Agile*; (iii) Conhecimento do negócio.

Respondendo às questões da investigação, é definida a solução a implementar:

1. Futuro processo de desenvolvimento de *software*;
2. Medidas a aplicar para que a adoção do *Scrum* se realize com sucesso.

Por último, são estabelecidas as principais definições para o projeto de implementação da solução, nomeadamente:

1. Orientações estratégicas para a implementação, focado nos seguintes aspetos: (i) Comunicação; (ii) Envolvimento das pessoas; (iii) Formação; (iv) *Coaching*; (v) Projetos piloto.
2. Organização do projeto baseado em dois níveis de decisão: (i) Grupo de Liderança; (ii) Grupo Operacional, que inclui o Grupo de Trabalho e a Equipa de projeto.
3. Cronograma do projeto de implementação;
4. Custos do projeto;
5. Avaliação da implementação;
6. Valor acrescentado para a organização.

Os resultados obtidos permitem concluir o seguinte:

1. Relativamente ao futuro processo de desenvolvimento de *software*, foi desenhado um processo que se adequa à realidade da CGD, apresentando as vantagens do modelo *Scrum*.
2. O sucesso da adoção do modelo *Scrum* é influenciado positivamente pelos seguintes fatores:
    a. A comunicação e acompanhamento do processo de transição;
    b. O envolvimento da organização no desenho final do processo de desenvolvimento de *software*.
    c. Troca de experiências entre os pares;
    d. Formação de todos os intervenientes no processo;
    e. *Coaching* das equipas e pessoas envolvidas;
    f. Realização de projetos piloto.
3. No que respeita ao futuro, após a adoção do *Scrum*, será necessária a aplicação das seguintes medidas:
    a. Plano de formação contínua adequado para colaboradores internos e externos, focado no *Scrum*, nas técnicas de desenvolvimento de *software*, no aumento de competências sociais e no negócio;
    b. Grupo de Apoio/*Coaching* permanente.

# 2 DEFINIÇÃO DO CONTEXTO DO PROBLEMA

Este trabalho de investigação consiste num projeto empresa, focado na adoção do *Scrum* (Schwaber and Beedle, 2001) - modelo de processo de desenvolvimento de *software Agile* - ao processo de desenvolvimento de *software* do banco Caixa Geral de Depósitos (CGD).

O desenvolvimento de *software* é uma atividade complexa, caracterizada por tarefas e requisitos que apresentam um elevado grau de variabilidade ao longo da sua execução (Bohem and Turner, 2004). Os processos de desenvolvimento de *software* - conjuntos de atividades relacionadas que conduzem à realização de um produto de *software* (Paulk *et al.*, 1993) - encontram-se em permanente evolução, influenciados quer pela constante alteração tecnológica, quer pela alteração das necessidades dos utilizadores dos produtos de *software* (Nerur *et al.*, 2005). Na escolha do melhor modelo de processo de desenvolvimento de *software*, a comunidade de desenvolvimento de *software* divide-se entre os tradicionalistas e os "agilistas", reclamando para si a superioridade dos seus modelos (Nerur *et al.*, 2005). Os tradicionalistas defendem os processos de desenvolvimento de *software* baseados nos modelos Tradicionais, como o *Waterfall*, o Desenvolvimento incremental ou a Engenharia de *software* orientada para a reutilização (Sommerville, 2011). Os "agilistas" defendem os processos de desenvolvimento de *software* baseados nos modelos *Agile*, como o *Scrum*, o *Extreme Programming* (XP), o *Crystal*, o *Feature Driven Development* (FDD), o *Dynamic Systems Develompment Method* (DSDM) ou o *Lean*.

O movimento *Agile* (Beck *et al.*, 2001) surge no ano de 2001, como reação (Bohem, 2002) às desvantagens da utilização de modelos Tradicionais de desenvolvimento de *software*. Os modelos *Agile*, desde então, têm aumentado a sua popularidade (Nerur *et al.*, 2005), tornando-se cada vez mais importantes para um significativo número de organizações, desafiando os modelos Tradicionais de processos de desenvolvimento de *software* (Salo and Abrahamsson, 2008). Na última década, o modelo *Scrum* tornou-se no modelo *Agile* mais utilizado, de acordo com o estudo realizado pela Forrester Research, Inc (West *et al.*, 2011), no ano de 2011.

Estudos passados demonstram que as alterações aos processos de *software* representam uma alteração organizacional complexa e não podem ser alcançados apenas com a substituição de ferramentas e das tecnologias (Nerur and Mahapatra, 2001). Os desafios inerentes à adoção dos modelos *Agile* em organizações que utilizam modelos Tradicionais são enormes (Bohem and Turner, 2005), podendo ser categorizados como desafios de gestão, organização, pessoas, processo e tecnológicos (Nerur *et al.*, 2005). A literatura existente mostra-nos que estes desafios têm sido ultrapassados por organizações como a Intel (Greene, 2004), a Yahoo! (Benefield,

2008), a Microsoft (Begel and Nagappan, 2007), a Nokia (Laanti *et al*., 2011), a 3M (Moore *et. al*, 2007), ou a Ericsson (Paasivaara and Lassenius, 2014). Os resultados apresentados por estas organizações têm servido de incentivo para que outras organizações alterem dos seus processos de desenvolvimento de *software*.

No contexto de uma organização que assenta os seus processos de desenvolvimento de *software* sobre o modelo Tradicional, *Waterfall*, com a realização deste trabalho de investigação, dando um contributo para o mundo académico no que respeita ao tema da adoção de modelos de desenvolvimento de *software* em organizações e contribuindo, de modo particular, para a aplicação do modelo *Scrum* na CGD, pretende-se que sejam obtidas respostas para as seguintes questões:

1. Qual deverá ser o desenho do futuro processo de desenvolvimento de *software*?
2. Que medidas deverão ser realizadas para que a adoção do *Scrum* ocorra com sucesso na organização?

# 3 REVISÃO DE LITERATURA

## 3.1 Processos de desenvolvimento de *software*

Um processo de desenvolvimento *software* pode ser definido como um conjunto de atividades relacionadas que conduzem à produção de um produto de *software* (Paulk *et al.,* 1993). Cada organização, com base no seu histórico, na sua estratégia e no tipo de projetos que desenvolve, adota determinado processo de desenvolvimento de *software*. Embora as atividades realizadas em cada processo de desenvolvimento de *software* variem de organização para organização, de forma geral, todos incluem quatro atividades em comum, fundamentais para a engenharia de *software*, nomeadamente: a especificação, o desenho e implementação, a validação e a evolução, do *software* (Sommerville, 2011), descritas na Tabela 1.

**Tabela 1 - Atividades fundamentais para a engenharia de *software***

Fonte: Sommerville (2011)

| Atividade | Descrição |
|---|---|
| Especificação do *software* | Atividade na qual são definidas as funcionalidades do produto de *software* a desenvolver, de acordo com o cliente do produto de *software*. São também identificados os constrangimentos operacionais espectáveis. |
| Desenho e implementação do *software* | Nesta atividade é realizado o desenho técnico do produto de *software* a desenvolver. Em conjunto com a Especificação de *software*, realizada na atividade anterior, permite a criação do produto de *software*. |
| Validação do *software* | Após a criação/desenvolvimento do produto de *software* é assegurado que as funcionalidades produzidas se encontram de acordo com o que foi definido na atividade de Especificação do *software*. Normalmente são realizados testes ao produto de *software* que garantem a sua qualidade de acordo com o desenho técnico realizado. |
| Evolução do *software* | Após a entrega do produto de *software* ao cliente, poderá existir a necessidade de evolução do mesmo ao longo do tempo, de acordo com as necessidades provenientes da sua utilização. |

Os processos de *software* podem ser classificados em duas categorias: processos Tradicionais e processos *Agile* (Bohem and Turner, 2003). Os processos Tradicionais são preditivos, no pressuposto que a informação sobre o produto de *software* a desenvolver é totalmente obtida antecipadamente. Os processos *Agile*, são adaptativos, assumindo que irão sempre existir necessidades de alteração durante o desenvolvimento do produto de *software* (Jammalamadaka and Krishna, 2013).

Embora cada organização siga o seu processo de desenvolvimento de *software* específico, cada abordagem individual geralmente apoia-se num modelo de processo de desenvolvimento de *software* Tradicional ou num modelo de processo *Agile* (Sommerville, 1996). Os modelos de processo de desenvolvimento de *software* são uma representação abstrata de um processo de desenvolvimento de *software*, fornecendo uma descrição do processo de uma perspetiva particular (Sommerville, 2011). Tal como os processos de *software*, também eles se dividem em duas categorias: modelos de processo Tradicionais e modelos de processo *Agile*, os quais se encontram seguidamente explicados.

### 3.1.1 *Modelos Tradicionais*

Os modelos de processos de *software* Tradicionais abordados neste trabalho de investigação, são os modelos genéricos, de acordo com Sommerville (2011): *Waterfall*, Desenvolvimento Incremental e Engenharia de *Software* Orientada para a Reutilização. Estes modelos genéricos, detalhados seguidamente, podem ser usados para explicar diferentes abordagens ao desenvolvimento de *software.*

#### *3.1.1.1 Waterfall*

A primeira publicação sobre o modelo *Waterfall* foi realizada por Wiston W. Royce no ano de 1970 (Royce, 1970). A abordagem surge da necessidade de obtenção de controlo na gestão de grandes desenvolvimentos de *software* (MacCormack, 2001). Trata-se do mais conhecido exemplo de um modelo de processo de desenvolvimento de *software* baseado num plano (*plan-driven*) no qual todas as atividades do processo de desenvolvimento de *software* deverão ser planeadas e agendadas antes do processo se iniciar (Jammalamadaka and Krishna, 2013). O modelo possui um processo bastante estruturado e sequencial, representado na Figura 1 (Sommerville, 2011). Sendo também conhecido por ciclo de vida do *software*, deve o seu nome *Waterfall* ao efeito visual de cascata criado pela sua sequência de atividades.

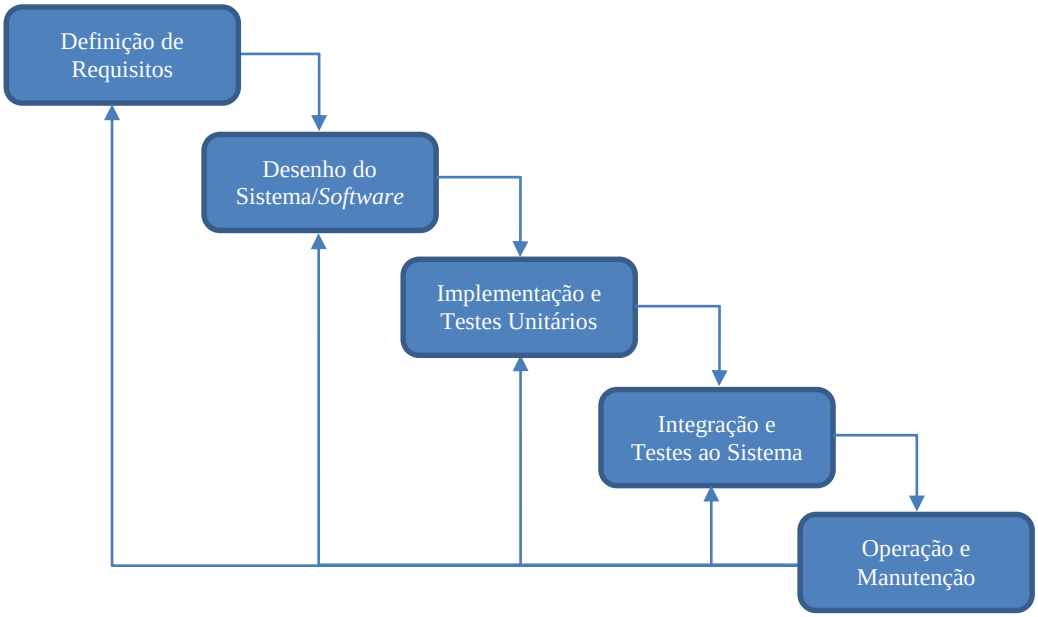


**Figura 1 - Modelo *waterfall***

Fonte: Sommerville (2011)

Divide-se em cinco fases, as quais devem ser completadas sequencialmente, com o objetivo de desenvolver um produto de *software*. Na Tabela 2 são explicadas as cinco fases sequenciais do modelo *Waterfall*: (i) Análise e definição de requisitos, (ii) Desenho do sistema e do *software*, (iii) Implementação e testes unitários, (iv) Integração e testes do sistema, (v) Operação e manutenção (Sommerville, 2011).

**Tabela 2 - Fases do modelo de processo de *software Waterfall***

Fonte: Sommerville (2011)

| Fase | Descrição |
|---|---|
| Análise e definição de requisitos | Primeira fase do modelo *Waterfall*. Os requisitos - serviços, funcionalidades, constrangimentos e objetivos do produto de *software* - são obtidos a partir dos clientes/utilizadores finais do produto de *software*. A informação recolhida é detalhada e serve para a especificação do produto de *software*, obrigatória para que se dar início à fase de Desenho do sistema e do *software*. |
| Desenho do sistema e do *software* | Com base na informação recolhida da fase anterior, a partir dos requisitos de *hardware* e de *software*, é definida e documentada a arquitetura global do sistema a realizar, |

| Fase | Descrição |
|---|---|
| | definindo-se desta forma o desenho técnico do produto de *software*. |
| Implementação e testes unitários | Nesta fase, a equipa de desenvolvimento desenvolve as funcionalidades pretendidas com base no desenho do produto de *software* realizado na fase anterior. Durante o desenvolvimento é construído o produto de *software*. A partir dos testes unitários verifica-se se cada peça do produto de *software* se encontra de acordo com o solicitado no desenho do mesmo. |
| Integração e testes do sistema | As várias peças do produto de *software* são integradas e testadas como um sistema completo, assegurando-se que os requisitos do *software* foram cumpridos. Caso não exista conformidade, o produto deverá ser corrigido. No final desta fase, o produto de *software* é entregue ao cliente. |
| Operação e manutenção | Normalmente, embora não necessariamente, esta é a fase mais longa deste ciclo de vida. O produto de *software* é instalado e colocado em utilização. A manutenção envolve a correção de erros que não são descobertos numa fase inicial do ciclo de vida, melhorando as unidades do produto de *software* e reforçando as funcionalidades com os novos requisitos que são identificados. |

O modelo *Waterfall* foi o primeiro a ser aplicado na maioria das organizações que realizam desenvolvimento de *software*, tendo sido ao longo do tempo, experimentados os seguintes problemas na sua utilização (Petersen *et al*., 2009):

- Elevado esforço na realização e aprovação da documentação;
- Elevado custo de mudança, no que respeita a alterações ao plano traçado;
- Entrega de valor para o cliente apenas no final do processo;
- Problemas são empurrados para as fases mais tardias do processo;
- Pouca transparência do processo e fraco envolvimento do cliente;
- Integração das peças de *software* e respetivos testes somente no final.

#### 3.1.1.2 Desenvolvimento Incremental

O modelo de desenvolvimento incremental começa a dar os primeiros passos nos anos 30, a partir do trabalho de Walter Shewhart, um especialista em qualidade na empresa Bell Labs, o qual propôs o ciclo de melhoria da qualidade, "Plan-Do-Study-Act" (PDSA) (Shewhart, 1986). Nos anos 40, o guru em qualidade W. Edwards Deming promoveu vigorosamente o PDSA, o qual mais tarde descreve em 1982 no livro "Out of the Crisis" (Deming, 1982). Trata-se de uma estratégia de desenvolvimento na qual as várias partes do produto são criadas separadamente e integradas quando completadas (Cockburn, 2008).

O modelo de desenvolvimento incremental é baseado na ideia de desenvolver um produto inicial, expondo-o aos comentários dos utilizadores e de clientes, e envolvendo-os nas várias versões do produto até este se adequar às necessidades pretendidas (Trammell *et. al*, 1996). Combina elementos do modelo *Waterfall* aplicando-o de maneira iterativa.

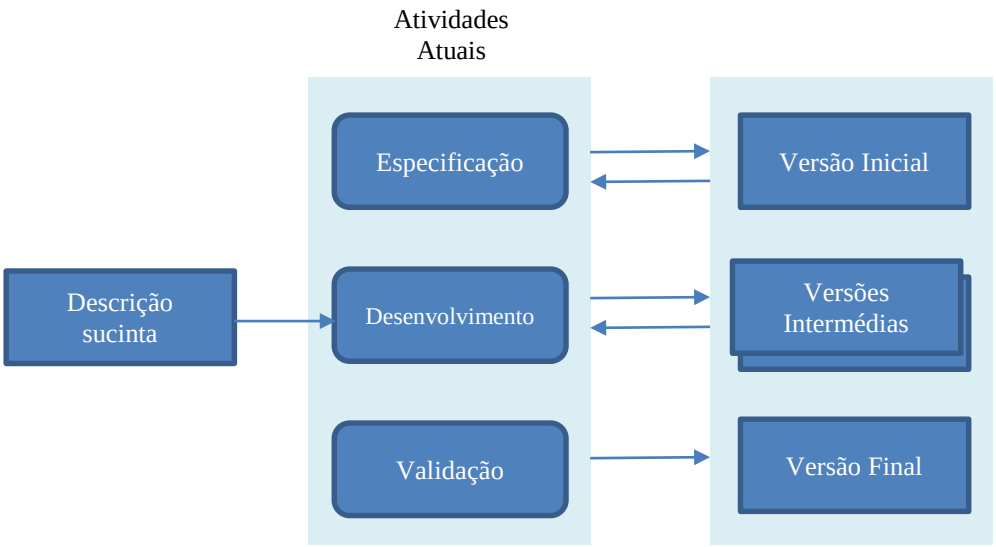


**Figura 2 - Desenvolvimento incremental**

Fonte: Sommerville (2011)

A Figura 2 apresenta as atividades do desenvolvimento incremental: (i) Especificação, (ii) Desenvolvimento e (iii) Validação, intervaladas por um *feedback* rápido dos utilizadores em cada atividade. Cada incremento ou versão do produto a ser realizado incorpora algumas das funcionalidades pretendidas pelos utilizadores. Geralmente, os incrementos iniciais incluem as funcionalidades mais importantes ou mais urgentes. Caso as funcionalidades desenvolvidas não satisfaçam as necessidades dos utilizadores, as alterações às mesmas são incorporadas nos próximos incrementos.

O modelo de desenvolvimento incremental tem três benefícios comparativamente ao modelo *Waterfall* (Sommerville, 2011):

- **Custo da mudança é reduzido** - Caso os requisitos se alterem, a quantidade de análise e da documentação a ser realizada é bastante menor que no modelo *Waterfall*;
- **É mais fácil obter o *feedback* do utilizador ou cliente** - O *feedback* é dado nas demonstrações das novas funcionalidades produzidas em cada incremento. É bastante difícil dar *feedback* sobre documentos de desenho de *software*, como acontece no modelo *Waterfall*;
- **Maior rapidez na entrega de valor para o utilizador** - As funcionalidades podem não estar todas produzidas, mas as principais podem já estar a ser utilizadas.

Do ponto de vista da gestão, os modelos de desenvolvimento incremental apresentam dois problemas (Sommerville, 2011):

- **O processo é pouco visível** - Sendo o desenvolvimento incremental é muito rápido e não sendo produzida documentação que reflita todas as versões do produto, não transmite informação regular que permita uma medição de progresso por parte da gestão.
- **A estrutura do sistema que está a ser desenvolvido tende-se a degradar em cada incremento** - Com pouco tempo e dinheiro a serem gastos na melhoria do *software*, a sua estrutura com alterações regulares tende a degrada-la. Incorporar alterações futuras vai sendo cada vez mais difícil.

#### 3.1.1.3 Engenharia de *software* orientada para a reutilização

A reutilização de *software* é a utilização de *software* existente para construção de novo *software* (Franks and Kang, 2005). O tipo de artefactos de *software* que podem ser reutilizados não se encontra limitado a fragmentos de código, mas podem incluir estruturas de desenho, especificações, documentação, entre outros (Freeman, 1983). A prática da reutilização de *software* pode ser realizada de duas formas (Brugali and Scandurra, 2009):

- **Oportunista** - A equipa de engenharia de *software* reutiliza peças de *software* que se adaptam e resolvem o problema atual.
- **Sistemática** - A equipa de investigação e desenvolvimento investe esforço no desenvolvimento de blocos de *software* que vão ao encontro das necessidades de uma família de produtos similares e que podem ser reutilizados para resolver uma abrangente classe de problemas.

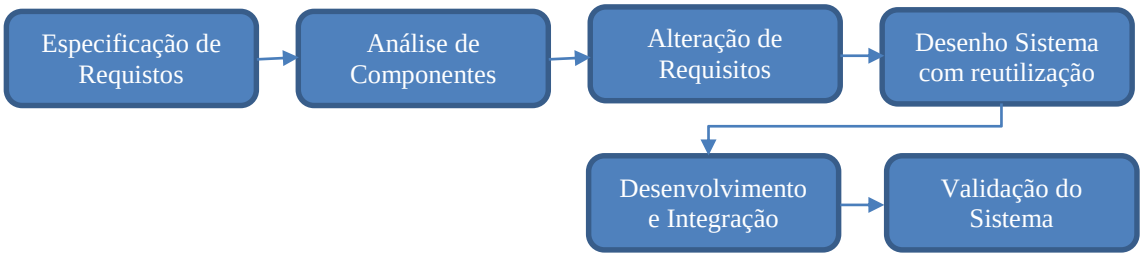


**Figura 3 – Engenharia de *Software* orientada à reutilização**

Fonte: Sommerville (2011)

O modelo de processo geral para o desenvolvimento baseado na reutilização pode ser consultado na Figura 3. Embora a especificação de requisitos e a validação possam ser comparados com outros modelos de processos de *software*, os passos intermédios são diferentes. A Tabela 3 descreve os vários passos do processo de desenvolvimento baseado na reutilização, de acordo com Sommerville (2011).

**Tabela 3 - Fases do modelo de engenharia de *software* orientada à reutilização**

Fonte: Sommerville (2011)

| Fase | Descrição |
|---|---|
| Especificação de requisitos | Esta fase é idêntica à fase de Análise e definição de requisitos do modelo *Waterfall* e à fase de Especificação do modelo incremental. |
| Análise de componentes | Após a fase de especificação de requisitos, é realizada uma pesquisa por componentes que satisfaçam essa especificação. Geralmente, não existe uma correspondência exata e os componentes que podem ser utilizados apenas fornecem uma parte das funcionalidades pretendidas. |
| Alteração de requisitos | Durante esta fase, os requisitos são analisados utilizando a informação sobre os componentes que foi descoberta previamente. Os requisitos são modificados para que reflitam as funcionalidades dos componentes disponíveis. Quando as modificações aos requisitos são impossíveis, tem que se voltar à fase anterior para procurar soluções alternativas. |
| Desenho do sistema com reutilização | Nesta fase é desenhado o produto de *software* tendo em conta os componentes reutilizados. Quando não existem |

| Fase | Descrição |
|---|---|
| | componentes reutilizáveis para determinada funcionalidade do novo produto de *software*, este tem que ser desenhado de raiz. |
| Desenvolvimento e integração | Os componentes de *software* que não conseguem ser externamente encontrados são desenvolvidos e integrados com os componentes existentes com o objetivo de criar o novo produto de *software*. A integração das componentes do produto de *software*, neste modelo, faz parte da atividade de desenvolvimento. |
| Validação do sistema | Esta última fase é idêntica à fase de integração e testes do sistema do modelo *Waterfall* e à fase de validação do modelo de Desenvolvimento Incremental. |

O modelo de Engenharia de *software* orientada à reutilização tem as seguintes vantagens (Sommerville, 2011):

- **Redução da quantidade de *software* desenvolvido** – devido à reutilização de componentes de *software*;
- **Redução dos custos** – dado que carece de menos desenvolvimento de coisas novas;
- **Redução dos riscos** – os componentes já se encontram a ser utilizados por outros *software*s;
- **Entregas mais rápidas de *software***.

Relativamente aos problemas na utilização do modelo, destacam-se os seguintes (Sommerville, 2011):

- **Cliente/Utilizador final pode não obter exatamente o que pretende** – a adaptação dos requisitos aos componentes utilizados pode levar a que o utilizador final não obtenha exatamente o que pretende;
- **Perda de controlo sobre o produto de *software*** - A organização que detém o sistema perde algum controlo sobre o mesmo quando depende das novas versões/evoluções dos componentes que não são seus mas que os estão a utilizar.

Explicados os modelos genéricos Tradicionais, *Waterfall*, Desenvolvimento Incremental e Engenharia de *Software* Orientada para a Reutilização, são apresentados na próxima secção os modelos *Agile*.

### 3.1.2 *Modelos Agile*

No ano de 2001, surge o movimento *Agile* (Beck *et al.*, 2001), como reação (Bohem, 2002) às desvantagens da utilização de modelos Tradicionais de desenvolvimento de *software* (Petersen *et al.*, 2009), como é o caso do *Waterfall*. Neste mundo competitivo, encontrando-se a tecnologia e a indústria em constante crescimento e as necessidades das organizações em constante alteração, os modelos de desenvolvimento *Agile* surgem com o objetivo de apoiar estas mudanças eficazmente (Highsmith and Cockburn, 2001). Quinze representantes de seguidores de diferentes modelos de desenvolvimento de *software,* tais como *Extreme Programming*, *Scrum*, DSDM, Crystal, Desenvolvimento orientado à funcionalidade, *Lean* e outros adeptos da necessidade de um modelo alternativo de desenvolvimento de *software,* juntaram-se para assinarem um manifesto (Beck *et al.*, 2001) para o desenvolvimento *Agile* de *software,* no qual pretendiam valorizar:

- **Indivíduos e interações** em vez de processos e ferramentas;
- ***Software* a funcionar** em vez de documentação;
- **Colaboração do cliente** em vez de negociação de contratos;
- **Resposta à mudança** em vez de seguir um plano.

Cumprindo os valores do manifesto *Agile*, os modelos *Agile* baseiam-se no modelo de desenvolvimento incremental, no qual são realizados pequenos incrementos, disponibilizando normalmente novas funcionalidades ao cliente a cada duas ou três semanas. A Tabela 4 apresenta os principais modelos *Agile* e uma breve descrição sobre os mesmos.

**Tabela 4 - Descrição dos principais modelos *Agile***

Fonte: Dyba and Dingsøyr (2008)

| Modelo *Agile* | Descrição |
|---|---|
| *Crystal* | Trata-se de uma família de modelos de processo de desenvolvimento de *software* para equipas que trabalham fisicamente juntas, de diferentes tamanhos e que realizam atividades de diferentes graus de criticidade. De acordo com a criticidade do sistema desenvolvido classificam-se como: Sem criticidade (*Clear)*, Amarela, Laranja, Vermelha, Azul. O modelo mais conhecido, o *Cristal Clear*, foca-se na comunicação de pequenas equipas a desenvolver *software* não crítico (*life-critical*). O desenvolvimento *Clear* tem sete características: entregas frequentes, |

| Modelo *Agile* | Descrição |
|---|---|
| | melhoria contínua com base na reflexão da equipa, comunicação cara a cara, os elementos da equipa podem partilhar o que pensam sem consequências negativas (segurança pessoal), foco, fácil acesso a utilizadores experientes e os requisitos para o ambiente técnico (testes automáticos, por exemplo). |
| *Dynamic Systems Development Method* (DSDM) | Divide os projetos em três fases: pré-projeto, ciclo de vida do projeto e pós-projeto. Assenta em nove princípios: envolvimento do utilizador, capacitar a equipa de projeto para realizar o seu trabalho, entregas frequentes, responder às necessidades de negócio, desenvolvimento iterativo e incremental, permitir reverter alterações, o âmbito de alto nível a ser fixado antes do início do projeto, testar ao longo do ciclo de vida e uma comunicação eficiente e eficaz. |
| *Feature Driven Development* (FDD) | Combina o *Model Driven Development* (MDD) – modelo que visa o desenvolvimento de produtos de *software* de forma rápida, eficaz e com baixo custo – e o desenvolvimento *Agile* com foco no modelo de objeto inicial, divisão do trabalho em funcionalidades e desenho iterativo para cada funcionalidade. Reivindica ser adequado para o desenvolvimento de sistemas críticos. Uma iteração de uma funcionalidade consiste em duas fases: Desenho e Desenvolvimento. |
| *Lean* | Trata-se de uma adaptação de princípios de produção *Lean*, em particular, do sistema de produção da Toyota, ao desenvolvimento de *software*. É composto por sete princípios: eliminar o desperdício, amplificar aprendizagem, decidir o mais tarde possível, entregar o mais rápido possível, capacitar a equipa, construir integridade, e ver o todo. |
| *Scrum* | Centra-se na gestão de projetos, em situações para as quais é difícil de planear com antecedência, com mecanismos de "controlo de processos empíricos", nos quais os ciclos de *feedback* constituem o elemento central. O *software* é desenvolvido em incrementos, chamados "*Sprints*", por uma equipa auto-organizada, começando com o planeamento e terminando com uma avaliação. As funcionalidades a serem implementadas são registados no artefacto *Backlog*. De seguida, o *Product Owner* decide quais itens do *Backlog* que devem ser |

| Modelo *Agile* | Descrição |
|---|---|
| | desenvolvido no seguinte *Sprint*. A equipa organiza as suas atividades numa reunião diária de pé. Um membro da equipa, o *Scrum Master*, é responsável por resolver os problemas que impedem a equipa de trabalhar de forma eficaz. |
| *Extreme Programming* (XP, XP2) | Concentra-se em melhores práticas para o desenvolvimento. Consiste em doze práticas: o jogo do planeamento, pequenas entregas, metáfora, desenho simples, testes, *refactoring*, programação em pares, propriedade coletiva, integração contínua, 40 horas por semana, os clientes no local e padrões de código. A revista "XP2" consiste nas seguintes práticas principais: sentar-se juntos, a equipa como um todo, espaço de trabalho informativo, o trabalho energizado, programação em pares, histórias, ciclo semanal, ciclo trimestral, folga, compilação de 10 minutos, integração contínua, programação a iniciar-se pelos testes e desenho incremental. |

De todos os modelos *Agile*, o modelo *Scrum*, é o mais utilizado de acordo com o estudo realizado pela Forrester Research, Inc, em 2011 (West *et al.*, 2011), o qual será aprofundado seguidamente neste trabalho de investigação.

### *3.2* Modelo *Scrum*

Hirotaka Takeuchi e Ikujiro Nonaka descreveram no seu artigo "The New New Product Development Game" de 1986, um novo modelo que aumentava a velocidade e a flexibilidade do desenvolvimento de novos produtos. Comparavam o novo método, no qual as fases possuíam forte intersecção e todo o processo era desenvolvido por equipas multifuncionais, com a modalidade desportiva, *rugby*, em que toda a equipa trabalha em conjunto para avançar no terreno e alcançar o objetivo comum. Nos anos 90, Ken Schwaber na sua empresa, a Advanced Development Methods, e Jeff Sutherland na Easel Corporation iniciaram a utilização deste modelo de desenvolvimento de *software*, sendo Jeff Sutherland o primeiro a utilizar o nome *Scrum* para o desenvolvimento de *software*.

O *Scrum* é um processo de desenvolvimento iterativo e incremental para o desenvolvimento de *software*. É um modelo aberto e imprevisível (Schwaber, 2001) que oferece um conjunto de ferramentas que dão visibilidade aos problemas existentes no processo de desenvolvimento. A literatura mostra-nos que o *Scrum* já foi adotado por grandes empresas

como a Intel (Greene, 2004), a Yahoo! (Benefield, 2008), a Microsoft (Begel and Nagappan, 2007), a Nokia (Laanti et al., 2011) a 3M (Moore et. al, 2007) ou a Ericsson (Paasivaara and Lassenius, 2014). Para o entendimento do modelo do *Scrum* é necessária a compreensão de alguns conceitos, seguidamente explicados: os Artefactos, os Eventos, e os Papéis e Responsabilidades.

### 3.2.1 *Artefactos*

Os artefactos do *Scrum*, descritos na Tabela 5 (Schwaber and Sutherland, 2013), nomeadamente o *Backlog* do produto, o *Backlog* do *Sprint* e o Incremento, representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação.

**Tabela 5 - Artefactos do *Scrum***

Fonte: Schwaber and Sutherland (2013)

| Artefacto | Descrição |
|---|---|
| *Backlog* do produto | Trata-se da lista de requisitos/necessidades fornecidos pelos clientes, equipa, gestores e utilizadores, mantida e priorizada pelo *Product owner* (papel do *Scrum*). |
| *Backlog* do *Sprint* | O *Backlog* do *Sprint* é um subconjunto do *Backlog* do produto, o qual contém os requisitos/necessidades que fazem parte de determinado *Sprint.* |
| Incremento | O incremento é a soma de todos os requisitos/necessidades do *Backlog* do produto, que foram disponibilizados no último *Sprint*, somado aos incrementos dos *Sprints* anteriores. No final do *Sprint*, o novo incremento deve encontrar-se totalmente finalizado de acordo com o planeado e deverá ser utilizável. |

### 3.2.2 *Eventos*

São identificados os seguintes eventos no *Scrum* (Schwaber e Sutherland, 2013): *Sprint*, Planeamento do *Sprint*, Reunião diária, Revisão do *Sprint e* Retrospetiva do *Sprint*. Todos os eventos são eventos *time-boxed*, isto é, todos os eventos têm uma duração máxima, a qual deverá ser cumprida com rigor. Os eventos encontram-se explicados na Tabela 6 abaixo.

**Tabela 6 - Eventos do *Scrum***

Fonte: Schwaber and Sutherland (2013)

| Eventos | Descrição |
|---|---|
| *Sprint* | O *Sprint* é o coração do *Scrum*. Durante o *Sprint* é criada uma versão incremental potencialmente utilizável do produto. A sua duração que pode ir de uma semana a um mês. Um novo *Sprint* é iniciado imediatamente após a conclusão do *Sprint* anterior. Os *Sprints* são compostos por uma reunião de planeamento do *Sprint*, Reuniões diárias, o trabalho de desenvolvimento, uma Revisão do *Sprint* e uma Retrospetiva do *Sprint*.<br>Durante o *Sprint* devem-se cumprir as seguintes regras:<br>• Não podem ser realizadas mudanças que possam colocar em perigo o objetivo do *Sprint*;<br>• As metas de qualidade não podem diminuir;<br>• O âmbito pode ser clarificado e renegociado entre o *Product owner* e a equipa de desenvolvimento. |
| Planeamento do *Sprint* | A reunião de planeamento do *Sprint* permite o planeamento do trabalho a ser realizado. Este plano é realizado pela equipa de desenvolvimento de forma colaborativa. Tem a duração máxima de 8 horas para *Sprints* de um mês. Para *Sprints* com durações menores que um mês, esta reunião deverá demorar menos tempo. O *Scrum master* tem um papel fulcral no cumprimento destes tempos. |
| Reunião diária | A Reunião diária tem duração máxima de quinze minutos, é realizada de pé e cada elemento da equipa responde a três questões:<br>1. O que fiz ontem?<br>2. O que vou fazer hoje?<br>3. Quais os meus impedimentos?<br>Esta reunião tem como objetivo a comunicação entre os elementos da equipa e a inspeção do progresso do *Sprint* face ao seu objetivo. |
| Revisão do *Sprint* | A reunião de Revisão do *Sprint*, realizada no final do *Sprint*, permite que a equipa apresente, ao cliente ou a qualquer parte interessada, o incremento de produto realizado. A partir do |

| Eventos | Descrição |
|---|---|
| | *feedback* dos presentes, adapta o *Backlog* do produto se necessário. O objetivo, com base no que foi produzido, é identificar o que poderá ser alterado nos próximos *Sprints*, de forma a otimizar o valor do produto para o cliente. Esta reunião não deve ultrapassar a duração de 4 horas para *Sprints* de um mês de duração, sendo este tempo adaptado conforme a duração do *Sprint*. |
| Retrospetiva do *Sprint* | Trata-se de uma oportunidade para a equipa olhar para ela mesma, identificando pontos de melhoria a aplicar no próximo *Sprint*. A equipa inspeciona-se a si mesma e adapta-se. Esta reunião, tal como a anterior, não deve ultrapassar a duração de 4 horas para *Sprints* de um mês de duração, sendo este tempo adaptado conforme a duração do *Sprint*. |

### 3.2.3 *Papéis e Responsabilidades*

Existem seis papéis no *Scrum*, de acordo com Schwaber e Beedle (2001), com diferentes tarefas e responsabilidades durante o processo: *Scrum master*, *Product owner*, Equipa de desenvolvimento, Cliente, Utilizador e a Gestão. Schwaber e Sutherland (2013) ignoram o Cliente, o Utilizador e a Gestão como papéis do *Scrum* e referem a existência apenas de 3 papéis fundamentais: *Scrum master*, *Product owner*, Equipa de desenvolvimento, os quais são detalhados na Tabela 7.

**Tabela 7 - Papéis e responsabilidades do *Scrum***

Fonte: Schwaber and Sutherland (2013)

| Papel | Descrição |
|---|---|
| *Scrum master* | O *Scrum master* é o responsável por assegurar a compreensão e o bom funcionamento do *Scrum* no seio da equipa. O *Scrum master* é um líder que tem que se disponibilizar para servir a equipa. Gere as interferências externas, ajudando os elementos externos à equipa a entender quais das suas interações com a mesma são úteis ou não. |

| Papel | Descrição |
|---|---|
| *Product owner* | O *Product owner* é o responsável por maximizar o valor do produto de *software* e do trabalho da equipa de desenvolvimento. É o elemento responsável por gerir o *Backlog* do produto transformando os requisitos dos clientes e da equipa, em funcionalidades do produto de *software*. Participa na estimativa do esforço de desenvolvimento. |
| Equipa de desenvolvimento | A equipa de desenvolvimento consiste num conjunto de profissionais que trabalham para entregar um "incremento de produto" em cada *Sprint*. As equipas de desenvolvimento devem ser auto-organizadas e compostas por elementos com conhecimentos transversais, necessários para o desenvolvimento do produto. |

O entendimento dos Artefactos, dos Eventos, e dos Papéis e responsabilidades do *Scrum*, torna mais fácil a compreensão do modelo dinâmico do *Scrum* apresentado na Figura 4, descrito da seguinte forma:

1. O elemento que desempenha o papel de *Product owner* recolhe as necessidades dos clientes, equipa de desenvolvimento, gestores e restantes *stakeholders*.
2. O *Product owner* tem a responsabilidade de criar e manter uma lista priorizada de requisitos recebidos, o artefacto *Backlog* do produto.
3. A Equipa de desenvolvimento realiza a reunião de Planeamento do *Sprint* na qual se compromete a realizar um conjunto de requisitos do *Backlog* do produto. Esses itens passam a fazer parte integrante do *Backlog* do *Sprint*, que contém as tarefas a realizar no âmbito do *Sprint*, assim como a indicação da duração do mesmo.
4. Dá-se início ao *Sprint* que corresponde a um ciclo de desenvolvimento com a duração de 1 a 4 semanas.
5. Durante o *Sprint* é realizada uma Reunião diária na qual cada elemento responde a 3 questões: o que fiz ontem, o que vou fazer hoje e quais os meus impedimentos.
6. No final do *Sprint* são realizadas duas reuniões: a Revisão do *Sprint*, na qual se apresentam a todos os interessados as funcionalidades desenvolvidas e a Retrospetiva do *Sprint*, na qual a equipa discute o que pode ser melhorado para conseguir realizar um melhor e mais rápido trabalho para o próximo *Sprint*.

7. São realizados os *Sprints* necessários para que o produto de *software* final fique concluído de acordo com as necessidades do cliente.

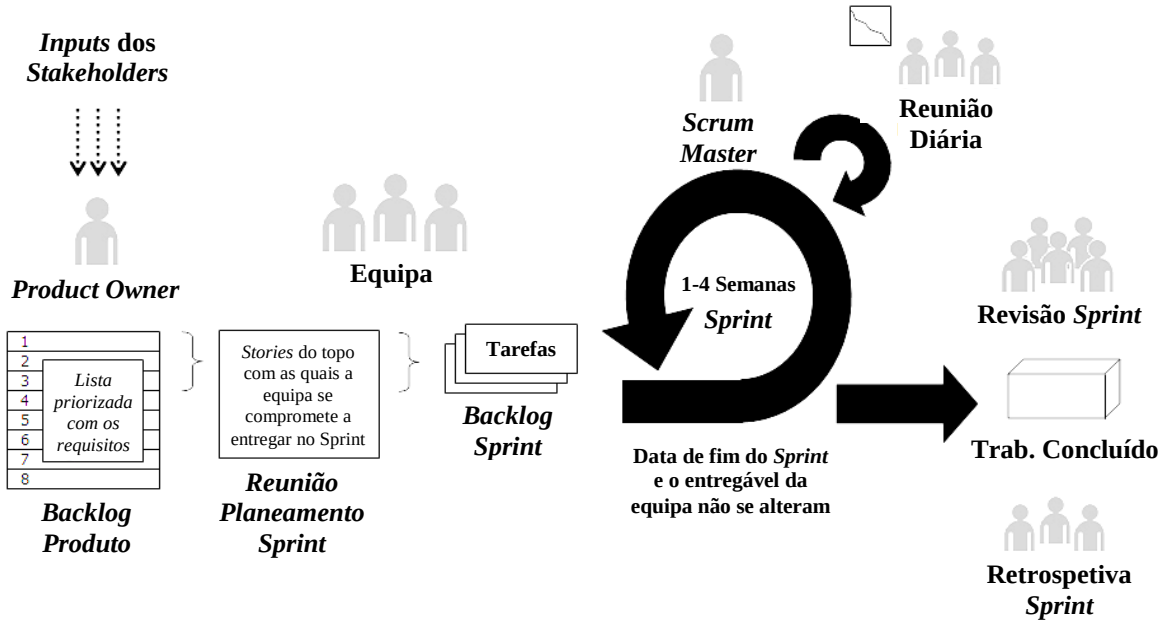


**Figura 4 – Modelo dinâmico do *Scrum***

Após a aplicação do modelo *Scrum* nas organizações, constata-se que muitas vezes este modelo tem que coexistir com o modelo existente na organização. A secção seguinte aborda as questões relativas a esta coexistência, em particular com o modelo Tradicional *Waterfall*.

## 3.3 Coexistência entre os modelos *Waterfall* e *Scrum*

Nas organizações, a maioria das implementações de modelos *Agile* são hibridas, misturando os modelos *Waterfall* e *Scrum* (West *et al.*, 2011). Os problemas encontrados na utilização em simultâneo do *Waterfall* e do *Scrum* diferem do ponto no qual se intersetam, sendo possíveis 3 cenários de interação (Sliger, 2006):

- ***Waterfall-up-front***. Neste cenário o *Waterfall* encontra-se no início do processo de desenvolvimento de *software*. As regras de governação das organizações requerem que sejam definidos os requisitos e os planos antes de se iniciar o trabalho de desenvolvimento. Em muitas organizações, estes planos formam o contrato entre o negócio e os sistemas de informação, o qual define a direção do projeto, a sua *timeline* e o seu orçamento. Após a aprovação do projeto, transita-se do modelo *Waterfall* para o modelo *Scrum*. Sliger (2006) recomenda que se siga o conselho de Alistair Cockburn (2000) e seja produzida a documentação estritamente necessária e suficiente, para esta fase.

- ***Waterfall-at-end***. Quando o *Scrum* e o *Waterfall* se encontram no fim do projeto, é geralmente na fase de testes do produto de *software*. Muitas vezes ocorre pela separação existente entre as equipas de desenvolvimento e as equipas de testes, nas organizações. A resposta a este cenário é dedicar um ou mais *Sprints* para completar os testes, incluindo as pessoas nesta forma *Agile* de trabalhar.
- ***Waterfall-in-tandem***. Esta talvez seja a mais difícil forma de interação entre o *Scrum* e o *Waterfall*. É o caso em que temos duas equipas a construir um produto, na qual uma funciona em *Scrum* e a outra em *Waterfall*. Os maiores problemas estão na coordenação do trabalho e na comunicação frequente. A equipa que trabalha em *Waterfall* prefere comunicar através de reuniões e documentos que definam totalmente a interface do que está a ser desenvolvido; A equipa de *Scrum* prefere deixar a interface menos definida e comunicar informalmente mas frequentemente as alterações à interface e comprometer-se progressivamente. Uma das forma de atenuar esta situação é convocar os gestores do projeto de *Waterfall* para as reuniões de Planeamento do *Sprint*.

West (2011) afirma que a maior parte das organizações que adotaram o *Scrum* encontram-se num modelo *Water-Scrum-Fall*:

- ***Water*.** Cenário idêntico ao *Waterfall*-up-front. West (2011) apresenta um conjunto de motivos para que não se desperdice tempo nesta primeira fase, nomeadamente: (i) Muitos requisitos iniciais são requisitos errados; (ii) Os utilizadores nem sempre conseguem comunicar o que pretendem; (iii) A equipa foca-se menos no cliente, dado que não é envolvida inicialmente; (iv) A equipa é menos multifuncional.
- ***Scrum***. Após a aprovação do projeto é adotado o modelo *Scrum* ao processo de desenvolvimento.
- ***Fall***. Entregas frequentes de *software* ao cliente possibilitam rápido *feedback* e asseguram que a entrega de valor é realizada o mais cedo possível. No entanto, a maioria das organizações não suportam entregas dinâmicas e flexíveis. Pelo contrário, realizam entregas não frequentes suportadas por um processo pesado. As equipas devem empurrar as fronteiras do processo de entrega: (i) Trazendo as equipas operacionais e o desenvolvimento para um trabalho mais próximo, trabalhando em parceria; (ii) Examinando e removendo obstáculos ao processo de entrega; (iii) Adicionando mais atividades de entrega aos *Sprint*s, caminhando para a automatização; (iv) Construindo objetivos partilhados conduzidos pelo negócio.

## 3.4 Transição dos modelos Tradicionais para os modelos *Agile*

Estudos passados demonstram que as alterações aos processos de desenvolvimento de *software* representam alterações organizacionais complexas e que não podem ser realizadas apenas substituindo as ferramentas e tecnologias atuais (Nerur *et al*., 2005). Estas mudanças podem ter impacto em vários aspetos da organização tais como na sua estrutura, na sua cultura, e nas práticas de gestão. Compreender o impacto deste fenómeno de mudança numa organização é o primeiro passo para a sua gestão e planeamento.

### 3.4.1 *Principais diferenças entre os modelos Tradicionais e Agile*

A transição de um processo baseado num modelo tradicional para um processo baseado num modelo *Agile*, implica um conjunto de alterações e adaptações ao processo existente, dada a grande diferença entre ambos os processos. Schuh (2004) e Nerur *et al.* (2005) identificaram e sistematizaram as principais diferenças entre os modelos Tradicionais e os modelos *Agile*. Conboy *et al.* (2010) complementaram essas diferenças, apresentando um conjunto de variáveis, apresentadas de forma sintetizada na Tabela 8.

Os modelos Tradicionais de desenvolvimento de *software* são centrados no processo, baseados no pressuposto de que as origens dos problemas e variações que ocorrem durante o ciclo de vida do desenvolvimento são identificáveis e podem ser eliminadas pelo refinamento e pela medição contínua dos processos (Cockburn and Highsmith, 2001a). O planeamento e o controlo acompanhados pelo estilo de gestão comando e controlo conduzem ao desenvolvimento de um produto de *software* (Highsmith, 2003). O modelo de ciclo de vida, como por exemplo o *Waterfall*, especifica as tarefas a serem realizadas e o que deverá ser entregue em cada fase, e atribui os papéis (tais como analistas, programadores, entre outros) a quem vai realizar estas tarefas (Nerur *et al.*, 2005). Adicionalmente ao produto de *software*, estes modelos introduzem uma grande quantidade de documentação que formalizam o conhecimento relativo ao processo e ao produto (Petersen *et al.*, 2009). A comunicação entre os participantes é também concretizada através destes documentos. Os clientes têm grande importância na especificação de requisitos, mas a sua participação é mínima no que respeita às restantes atividades (Sommerville, 2011). Estes modelos são apropriados tanto para tecnologias orientadas a objetos ou não orientadas a objetos (Nerur *et al.*, 2005).

Por outro lado, os modelos *Agile* lidam com a imprevisibilidade focando-se nas pessoas e na sua criatividade em detrimento do foco nos processos (Cockburn and Highsmith, 2001b). São caracterizados por ciclos iterativos de desenvolvimento orientados às funcionalidades do produto de *software*, períodos de reflecção e introspeção, poder de decisão colaborativo,

incorporação do rápido *feedback* e alteração, e integração contínua das alterações ao código no sistema em desenvolvimento (Schwaber and Sutherland, 2013). Os programadores trabalham em pequenas equipas com os clientes/utilizadores finais do produto de *software*, membros ativos da equipa. Os modelos *Agile* favorecem um estilo de gestão de liderança colaborativa na qual o papel de gestor de projeto é de facilitador ou de coordenador (Highsmith, 2003). A documentação é desencorajada, considerando-se que o código documenta o produto de *software* e o conhecimento encontra-se, tacitamente nas pessoas. A rotatividade da equipa assegura que este conhecimento não é monopolizado. Os modelos *Agile* são caracterizados por uma forte componente social na qual a colaboração e a comunicação as pessoas são a base para a ação coletiva (Cockburn and Highsmith, 2001b). A iteratividade dos modelos *Agile* é favorecida pelas tecnologias orientadas a objetos (Nerur *et al.*, 2005).

**Tabela 8 - Diferenças entre os modelos tradicional e o *Agile***

Fonte: Conboy *et al.* (2010)

| Componente | Tradicional | *Agile* |
|---|---|---|
| **Pressupostos** | Sistemas totalmente especificados, preditivos, e construídos de acordo com um plano meticuloso e extenso. | *Software* adaptativo e de alta qualidade, desenvolvido por pequenas equipas, utilizando os princípios de melhoria continua, baseados em testes com rápido *feedback* e alteração. |
| **Controlo** | Centrados no processo | Centrados nas pessoas |
| **Estilo de gestão** | Controlo e comando | Liderança e colaboração |
| **Gestão do conhecimento** | Explicito | Tácito |
| **Atribuição de papéis** | Individuais – favorece a especialização | Equipas auto-organizadas – encoraja a alteração de funções |
| **Comunicação** | Formal e quando necessário | Informal e contínua |
| **Papel do cliente** | Importante | Crítico |
| **Ciclo do projeto** | Orientado a tarefas ou atividades | Orientado a funcionalidades do produto |
| **Modelo desenvolvimento** | Modelo de ciclo de vida (*Waterfall* ou variações) | Modelo entrega evolutiva |

| Componente | Tradicional | *Agile* |
|---|---|---|
| **Organização/Estrutura desejável** | Mecânica (Burocrática com elevada formalidade) | Orgânica (Flexível e participativa encorajando ação social cooperativa) |
| **Tecnologia** | Sem restrição | Favorece tecnologias de desenvolvimento orientadas a objetos |
| **Localização equipa** | Predominantemente distribuída | Predominantemente co-alocada |
| **Tamanho equipa** | Geralmente maior que 10 elementos | Geralmente menor que 10 elementos |
| **Aprendizagem contínua** | Não é frequentemente encorajada | Abraçada |
| **Cultura de gestão** | Controlo e comando | Responsivo |
| **Participação da equipa** | Não obrigatória | Necessária |
| **Planeamento projeto** | No início | Continuo |
| **Mecanismos de *feedback*** | Não obtido facilmente | Geralmente vários disponíveis |
| **Documentação** | Substancial | Mínima |

### 3.4.2 *Formas de transição*

Existem diferentes formas de adoção dos modelos de processo de desenvolvimento de *software Agile*. Com base nas organizações que já realizaram a transição de modelos Tradicionais para modelos *Agile*, Cohn (2010), identificou quatro formas diferentes: (i) Transição *Start Small*, (ii) Transição *All In*, (iii) Transição pública e a (iv) Transição discreta.

**(i) Transição *Start Small*.** O conselho mais comum no que respeita à transição para um processo *Agile* tem sido começar com um projeto piloto, aprender com o mesmo e então disseminar o *Agile* faseadamente pela organização (Cohn, 2010). Segundo Cohn (2010), escolhe-se uma a três equipas (de cinco a nove elementos cada) para aplicar o novo modelo de processo de desenvolvimento de *software*. Depois de aplicado com sucesso expande-se gradualmente ao resto da organização. Durante a expansão, as novas equipas beneficiam das

lições aprendidas pelas equipas anteriores. Esta foi a forma utilizada com sucesso pela empresa Yahoo! no ano de 2005 (Benefield, 2008), numa das maiores implementações de *Agile* no mundo, envolvendo mais de 150 equipas de desenvolvimento, nos Estados Unidos, na Europa e na Asia.

O processo de transição na Yahoo! iniciou-se com um programa piloto com 4 equipas voluntárias a tentar utilizar o modelo *Scrum* e a partilhar as suas experiências para o resto da organização. O comprometimento inicial das equipas foi: (i) a compreensão completa do *Scrum*, traduzida na certificação "Certified *Scrum*Master" para a maior parte dos elementos das equipas; (ii) trabalhar com *coaches* de *Scrum* externos durante os primeiros *Sprint*s; (iii) utilizar todas as práticas *standard* do *Scrum* descritas no livro "*Agile* Project Management with *Scrum*" de Ken Shwaber's; (iv) completar pelo menos um *Sprint* de 30 dias. Ficou também claro que depois do primeiro *Sprint* as equipas podiam escolher abandonar o *Scrum* se chegassem à conclusão que não trazia benefícios.

Ao final de dois meses, após uma maioria de *feedback* positivo, outras equipas começaram a ficar interessadas na adoção do *Scrum*. Para demonstrar a toda a organização a experiência positiva das equipas envolvidas, foi realizado um questionário *online* e apresentados os respetivos resultados.

De acordo com Cohn (2010), as razões para a escolha de uma transição *Start Small* são as seguintes: (i) Custos menos elevados pois não carece de tantos *coaches* como na transição *All In*; (ii) Sucesso mais cedo quase garantido pela escolha das primeiras equipas a adotar o novo modelo; (iii) Evita o risco de se cometer o mesmo erro em toda a organização; (iv) Menos stressante; (v) Não carece de uma reorganização inicial da organização, poderá ser feita mais à frente no processo.

**(ii) Transição *All In.*** A transição *All In* é o oposto da transição *Start Small*. Tem como objetivo aplicar o novo modelo de processo de desenvolvimento de *software* a toda a organização ao mesmo tempo. Esta foi a forma utilizada com sucesso pela organização Salesforce.com, no ano de 2007, envolvendo duzentos colaboradores, numa janela temporal de três meses (Fry and Greene, 2007). Esta foi uma das maiores e mais rápidas transições *All In* a nível mundial. A organização Salesforce.com realizou 45 sessões de trabalho de uma hora com pessoas chave dos vários níveis da organização. Estas sessões serviram para a definição do novo processo e para a existência de um *buy*-in relativo à mudança.

De acordo com Cohn (2010), as razões para a escolha de uma transição *All In* são as seguintes: (i) Redução da resistência, pela rapidez com que a transição é realizada; (ii) evita os problemas da coexistência temporária dos dois modelos, o antigo e o novo; (iii) Rapidez.

**(iii) Transição pública.** O padrão de transição pública tem como objetivo comunicar a toda a organização ou para fora da organização, que esta iniciou a adoção de um novo modelo de processo de desenvolvimento de *software*.

Cohn (2010) apresenta as razões para tornar publica a adoção das metodologias *Agile*: (i) Todos sabem o que se está a fazer, e por isso existem mais possibilidade de adoção; (ii) Uma transição pública estabelece uma visão para ser perseguida; (iii) Demostra uma declaração firme do nosso compromisso; (iv) Pode-se solicitar ajuda à organização; (v) Anunciar o objetivo e atingi-los transmite uma mensagem poderosa.

**(iv) Transição discreta.** O oposto é a transição discreta, na qual apenas os elementos envolvidos na transição têm conhecimento que a mesma está a decorrer.

A privacidade foi identificada como muito importante pela Yahoo! (Benefield, 2008). Cometeram o erro de divulgar a toda a organização o nome das equipas e respetivos resultados, com vista à criação de competitividade. Para algumas equipas que não atingem o resultado esperado pode criar um sentimento oposto de desmotivação e vergonha.

As razões para a realização de uma transição discreta, segundo Cohn (2010) são: (i)

A oportunidade de realizar progresso antes da resistência começar; (ii) Manter afastada a pressão adicional; (iii) Ninguém saber até nós comunicarmos; (iv) Se ninguém souber, ninguém nos dirá para parar.

### 3.4.3 *Fatores de sucesso relativos às pessoas*

Cohn e Ford (2003) identificaram um conjunto de problemas que ocorrem na transição das organizações para processos *Agile*. Dividiram os problemas em duas categorias: problemas dos programadores e problemas dos recursos humanos em geral. Mais tarde, Bohem e Turner (2005) identificaram desafios de gestão na implementação de projetos *Agile* em organizações com processo de desenvolvimento Tradicionais. Colocando de parte os desafios técnicos, dividem os desafios nas categorias de processo de desenvolvimento, processo de negócio e pessoas. Nerur *et al.* (2005), com o seu estudo, acrescentam os desafios tecnológicos, criando quatro categorias: gestão e organização, pessoas, processos e tecnologia (ferramentas e técnicas). Chow e Cao (2008) apresentam um conjunto de fatores críticos de sucesso relativos

aos projetos *Agile*, divididos nas categorias de Organização, Pessoas, Processo e Técnicos. Misra *et al.* (2009) identificam os fatores de sucesso na adoção de práticas de *software Agile*, apresentando nove fatores de sucesso divididos entre pessoas e organização. Conboy *et al.* (2010) focam-se nos desafios das pessoas, identificando um conjunto de desafios e de recomendações para os ultrapassar. Cohn (2010) baseado na sua experiência no terreno com várias empresas internacionais apresenta o que deve ser feito para se ter sucesso numa implementação de *Agile*, em particular com *Scrum*.

Baseado na literatura acima referida, os fatores que influenciam a transição de processos baseados em modelos Tradicionais para processos baseados em modelos *Agile* podem ser classificados em quarto categorias: Organização, Pessoas, Processo e Tecnologia, apresentados na Tabela 9.

**Tabela 9 - Fatores de sucesso na adoção de modelos *Agile***

Fonte: Elaboração própria

| Dimensão | # | Fatores de Sucesso |
|---|---|---|
| **Organização** | 1 | Apoio e compromisso da gestão de topo |
| | 2 | Cultura organizacional |
| | 3 | Forma organizacional |
| | 4 | Estilo de gestão |
| | 5 | Gestão do conhecimento do desenvolvimento de *software* |
| | 6 | Localização das equipas e acessibilidade pelo cliente |
| | 7 | Política de contratação de recursos humanos |
| | 8 | Boa redefinição dos papéis e responsabilidades existentes |
| | 9 | Sistema de avaliação e recompensa adaptado ao modelo *Agile* |
| **Pessoas** | 10 | Aprendizagem e compreensão dos valores e os princípios *Agile* |
| | 11 | Elevadas competências e conhecimentos generalistas de desenvolvimento |
| | 12 | Elevadas competências sociais (comunicação) |
| | 13 | Elevado conhecimento do negócio |
| | 14 | Motivação elevada/Aceitação do novo modelo de desenvolvimento |
| | 15 | Trabalho em equipa |
| | 16 | Envolvimento constante do cliente |
| | 17 | Tomada de decisão descentralizada |

| Dimensão | # | Fatores de Sucesso |
|---|---|---|
| **Processo** | 18 | Seleção do método *Agile* apropriado |
| | 19 | Mudança da aproximação orientada ao processo para uma aproximação orientada às pessoas e funcionalidades |
| | 20 | Desenvolvimento curto, iterativo e orientado a testes que realça a adaptabilidade |
| | 21 | Gestão de projetos grandes e escaláveis |
| | 22 | *Standards* de processo, como CMMI ou ISO |
| | 23 | Requisitos formais |
| | 24 | Interface e integração com outras equipas que utilizam outros modelos |
| | 25 | Nível adequado de documentação |
| **Tecnologia** | 26 | Inadequação das tecnologias e ferramentas existentes |
| | 27 | Aplicação de técnicas *Agile* de desenvolvimento de *software* |

As pessoas são vitais na adoção dos modelos *Agile*, de acordo com Bohem e Turner (2005). Na transição de processos Tradicionais para processos *Agile*, destacamos neste trabalho de investigação, os fatores de sucesso que afetam diretamente as pessoas:

1. Motivação elevada/Aceitação do novo modelo de desenvolvimento;
2. Aprendizagem e compreensão dos valores e os princípios *Agile*;
3. Aumento generalizado das competências de desenvolvimento;
4. Aumento das competências sociais (facilitadoras da comunicação);
5. Aumento do conhecimento do negócio (facilitador da comunicação com cliente);
6. Trabalho em equipa, coerente e auto-organizado;
7. Envolvimento constante do cliente;
8. Tomada de decisão descentralizada.

**#1 Motivação elevada/Aceitação do novo modelo de desenvolvimento.** Uma mudança é composta por um aspeto técnico e um aspeto social (Lawrence, 1969). O aspeto técnico da mudança tem a ver com as alterações às rotinas físicas. O aspeto social da mudança tem a ver com a forma como os afetados pensam que esta irá alterar os seus já estabelecidos relacionamentos dentro da organização. Na maior parte das vezes apenas o aspeto técnico da mudança é determinante para a sua aceitação. Mas de facto o aspeto social é o que determina a

presença ou ausência de resistência. Os especialistas em gestão da mudança vulgarmente descrevem que os anticorpos organizacionais se começam a juntar a partir do momento que algo de novo aparece na cultura existente (Bohem and Turner, 2005).

As principais razões da resistência à mudança nas organizações, de acordo com Creasey e Hiatt (2007) são apresentadas na Tabela 10. Apresentam diferentes razões consoante o papel desempenhado na organização, funcionário ou gestor.

**Tabela 10 - Principais razões da resistência à mudança**

Fonte: Creasey e Hiatt (2007)

| # | Funcionários | Gestores |
|---|---|---|
| **1** | Falta de conhecimento sobre a situação | Medo de perder controlo e autoridade |
| **2** | Medo do desconhecido | Falta de tempo |
| **3** | Falta de segurança no trabalho | Conforto com o *status quo* |
| **4** | Falta de *sponsorship* | Sem resposta a "O que está aí para mim?" |
| **5** | | Não envolvimento no desenho da solução |

Para ultrapassar a resistência das pessoas e motiva-las para a utilização do nosso processo de desenvolvimento de *software*, Bohem e Turner (2005), Conboy et al. (2008) e Cohn (2010) recomendam um **plano de comunicação eficaz**, o qual deverá ter em conta as seguintes ações:

1. Gestão de topo deverá comunicar as razões da mudança;
2. Superior hierárquico direto deverá comunicar como esta mudança os deverá afetar;
3. Criar oportunidades para ouvir os seus pares dentro da organização;

E para além do plano de comunicação deverão ser realizadas as seguintes ações:

- **Envolver as pessoas no processo de transição**, principalmente os mais céticos;
- **Realizar pilotos com sucesso em diferentes tipos de projetos;**
- **Alinhar os incentivos existentes com os futuros;**
- **Tentar ter sempre pessoas "*bought-in*" em cada equipa.**

**#2 Aprendizagem e compreensão dos valores e os princípios *Agile*.** A utilização nas organizações de processos de *software* baseados em modelos *Agile*, carece da aprendizagem e compreensão dos valores e os princípios *Agile* por parte das pessoas. A incompreensão dos mesmos leva a que as equipas de desenvolvimento não atinjam os objetivos propostos pelos modelos de processo (Conboy *et al.*, 2010). Embora a formação em *Agile* possa ser vista como

uma solução típica para a aprendizagem das suas práticas, não é suficiente para que as equipas de desenvolvimento compreendam e apliquem os seus valores e princípios. A incompreensão destes princípios deve-se na maioria das vezes a formação disfuncional e inadequada (Gandomani *et al.*, 2014). Os motivos desta disfunção e inadequação vão desde a formação parcial das pessoas envolvidas, conteúdos formativos inadequados, formação focada na teoria, falta de formação contínua, até a aspetos do comportamento humano. Mahanti (2006) alerta para a dependência da variável tempo para que as pessoas interiorizem os valores e princípios dos modelos.

De acordo com o estudo realizado por Conboy *et al.* (2010), para além de uma **primeira formação à equipa**, deverão ser adotadas as seguintes medidas:

- **Formação contínua** - possibilitando aos seus colaboradores a participação em formações e conferências *Agile*, focadas nos seus valores e princípios;
- **Acompanhamento da equipa por um elemento com experiência (*coaching*) em processos de *software Agile*** - com o objetivo de que a equipa retenha os valores e princípios *Agile*. A inclusão de um elemento com experiência em modelos *Agile* a acompanhar a equipa, influencia positivamente a retenção dos valores e princípios por parte das equipas, de acordo com Gandomani *et al.* (2014).

**#3 Aumento generalizado das competências de desenvolvimento**. O desafio do trabalho em equipa, passa pela alteração do trabalho realizado por especialistas, de forma isolada, para um trabalho colaborativo e multifuncional (Marchenko and Abrahamsson, 2008). As fronteiras entre as responsabilidades dos elementos das equipas de desenvolvimento são menos claras nos modelos *Agile* do que nos modelos Tradicionais, sendo importante que ganhem competências no maior conjunto de áreas possível (Conboy *et al.*, 2010).

De acordo com Conboy *et al.* (2010) as seguintes medidas deverão ser implementadas de forma a ultrapassar o desafio:

- **Utilizar *pair programming* e *pair rotation*** - técnicas de desenvolvimento de *software* para partilhar o conhecimento e facilitar a aprendizagem;
- **Encorajar a autoatribuição de tarefas** - permitindo que os elementos das equipas trabalhem em diferentes áreas e adquiram novas competências;
- **Reintroduzir papéis específicos** - quando é entendido que traga algum benefício às equipas de grande dimensão ou com conflitos entre os seus elementos, por exemplo.

Mahanti (2006) acrescenta a **formação** como medida a adotar, para que as pessoas se tornem especialistas em várias áreas e tenham um entendimento dos aspetos técnicos e de negócio do desenvolvimento de *software*.

**#4 Aumento das competências sociais.** A comunicação realizada pelas equipas *Agile* assenta fortemente numa comunicação cara-a-cara. As práticas *Agile* tais como a partilha do local de trabalho, trabalhar diretamente com o cliente, reuniões diárias, reuniões de retrospetivas, são exemplos do aumento da interação social que carece de capacidade de sociabilização, comunicação e apresentação (Lalsing *et al.,* 2012). Conboy (2010) conclui que apesar da maioria das pessoas que estudou serem tecnicamente talentosas, são fracas em termos de comunicação e apresentação, propondo as seguintes medidas:

- **Formação em competências sociais** – utilizando nas mesmas exemplos concretos como as gravações das reuniões diárias para posterior discussão;
- **Utilização de documentação apropriada que facilite a comunicação** – mais utilizado quando a maioria dos elementos da equipa é inexperiente;

**#5 Aumento do conhecimento do negócio.** O desenvolvimento *Agile* implica uma constante interação entre o cliente e a equipa que desenvolve o produto de *software*. Esta interação carece de uma linguagem comum, baseada no conhecimento do negócio (Conboy *et al.*, 2010). Para que as equipas aumentem o seu conhecimento relativamente ao negócio, de acordo com Conboy *et al.* (2010), a organização deverá:

- **Realizar sessões de formação em tópicos básicos sobre o negócio**;
- **Disponibilizar módulos de formação** que permitam à equipa adquirir os conhecimentos de negócio para o seu projeto;
- **Recrutar colaboradores** que tenham uma combinação de conhecimentos em sistemas de informação e conhecimento do negócio.

**#6 Trabalho em equipa, coerente e auto-organizado.** O trabalho em equipa é o coração do processo de desenvolvimento *Agile* (Cohn, 2010). O trabalho em equipa e a construção da mesma são críticos para a existência de equipas auto-organizadas (Begel and Nagappan, 2007). Para a melhoria do trabalho em equipa, Cohn (2010) recomenda o seguinte:

- **Focar a responsabilidade na equipa** e não no individuo;

- **Realizar uma boa e moderada utilização dos especialistas** – As equipas devem ter elementos mais generalistas, que ao serem suportados por elementos especialistas, aumentam as suas competências em aspetos mais específicos do processo de desenvolvimento;
- **Paralelizar tarefas da equipa** – As equipas habituadas a processos de desenvolvimento sequencial, esperam que o trabalho de análise seja terminado para iniciar o desenho, de seguida a programação e somente no final os testes. O que se pretende é que se paralelize trabalho e se iniciem todas estas tarefas o mais cedo possível.
- **Aumentar e partilhar conhecimento constantemente**.

No que diz respeito à melhoria da auto-organização, Cohn (2010) indica algumas formas de influenciar a organização das equipas, introduzindo um conceito de ajuste de contentores. Os contentores são o limite dentro do qual a auto-organização da equipa ocorre. Cohn ajusta estes contentores, amplificando ou amortecendo as diferenças ou alterando as trocas nas quais os elementos das equipas participam. Não existe uma fórmula que resolva qualquer situação. Cabe aos líderes influenciarem a auto-organização das equipas através de um conjunto de ações apresentado na Tabela 11.

**Tabela 11 - Ações que influenciam a auto-organização das equipas**

Fonte: Cohn (2010)

| Categoria | Ações |
|---|---|
| **Ajustar os Contentores** | Alterar o número de elementos da equipa<br>Alterar quem faz parte da equipa |
| | Dar mais ou menos responsabilidade à equipa |
| | Alterar o espaço físico da equipa - dar mais ou menos espaço, todos juntos ou mais separados, etc. |
| **Amplificar ou amortecer as diferenças** | Introduzir um novo elemento na equipa com mais poder, experiência, etc. |
| | Realizar questões difíceis à equipa para assegurar que diferentes pontos de vista são ouvidos. |
| | Alterar o estilo de tomada de decisão da equipa |
| | Encorajar pontos de vista contrários |
| **Alterar trocas** | Adicionar ou remover pessoas de uma troca |

| Categoria | Ações |
|---|---|
| | Formalizar ou tornar menos formal uma determinada troca |
| | Alterar a forma como uma troca decorre (comunicação face a face, documentação) |
| | Alterar a frequência de uma troca |

**#7 Envolvimento constante do cliente.** O sucesso do desenvolvimento *Agile* depende dos clientes participarem ativamente no processo de desenvolvimento (Nerur *et al*., 2005). Para a existência deste envolvimento é necessário que o cliente esteja constantemente presente no processo de desenvolvimento.

**#8 Tomada de decisão descentralizada.** A tomada de decisão no modelo tradicional está concentrada no gestor de projeto, por outro lado, no modelo *Agile* está dispersa pela equipa. Para facilitar esta tomada de decisão, Conboy *et al*. (2010), recomendam as seguintes ações:

- **Construir um ambiente de partilha e aprendizagem** para dar à equipa capacidade de tomada de decisão;
- **Implementar um sistema democrático de voto**;
- **O gestor de projeto deverá desempenhar um papel de facilitador**.

# 4 QUESTÕES RESULTANTES DA REVISÃO DE LITERATURA

Neste capítulo são descritas as principais questões da revisão da literatura que serão posteriormente necessárias para a aplicação do *Scrum* ao processo de desenvolvimento de *software* da CGD. Este trabalho de investigação tem como objetivo a definição de um plano de aplicação do modelo *Scrum* (Schwaber and Beedle, 2001), modelo de processo de desenvolvimento de *software Agile*, ao processo de desenvolvimento de *software* do banco CGD, centrado nos desafios relativos às pessoas. Como tal será necessário ter em conta os conceitos de processo de desenvolvimento de *software* e modelo de processo de desenvolvimento de *software*.

Um **processo de desenvolvimento de *software*** é um conjunto de atividades relacionadas que conduzem à produção de um produto de *software* (Paulk *et al.*, 1993). Apesar da enorme variedade de atividades nos processos de *software*, todos eles incluem quatro atividades fundamentais para a engenharia de *software*: a especificação, o desenho e implementação, a validação e a evolução, do *software* (Sommerville, 2011). Podem ser classificados em duas categorias (Jammalamadaka and Krishna, 2013):

(i) **Processos Tradicionais** - são preditivos, no pressuposto que a informação sobre o produto de *software* a desenvolver é totalmente obtida antecipadamente.

(ii) **Processos *Agile*** - são adaptativos, assumindo que irão sempre existir necessidades de alteração durante o desenvolvimento do produto de *software*.

Os **modelos de processo de desenvolvimento de *software*** são uma representação abstrata de um processo de desenvolvimento de *software*, fornecendo uma descrição do processo de uma perspetiva particular (Sommerville, 2011). Tal como os processos de *software*, também eles se dividem em duas categorias: **modelos de processo Tradicionais** e **modelos de processo *Agile***. De entre os modelos de processo Tradicionais destacam-se os modelos (Sommerville, 2011): ***Waterfall***, Desenvolvimento Incremental, e Engenharia de *software* orientada para a reutilização. Dos modelos *Agile* destaca-se os modelos (Dyba and Dingsøyr, 2008): ***Scrum***, *Extreme Programming* (XP), *Crystal*, *Feature Driven Development* (FDD), *Dynamic Systems Develompment Method* (DSDM) e *Lean*.

Deste modo, importa aferir qual processo de desenvolvimento de *software* utilizado na CGD e em que modelo se baseia.

West (2011) afirma que a maior parte das organizações que adotaram o *Scrum* encontram-se num modelo hibrido de **Water-*Scrum*-Fall**. Os problemas encontrados na utilização em simultâneo do *Waterfall* e do *Scrum* diferem do ponto no qual se intersetam,

sendo possíveis 3 cenários de interação (Sliger, 2006): ***Waterfall*-up-front, *Waterfall*-at-end e *Waterfall*-in-tandem.** Na solução a adotar para a CGD estas questões deverão ser levadas em conta.

Existem diferentes formas de adoção dos modelos de processo de desenvolvimento de *software Agile*. Com base nas organizações que já realizaram a transição de modelos Tradicionais para modelos *Agile*, Cohn (2010), identificou quatro formas diferentes: (i) **Transição *Start Small***, (ii) **Transição *All In***, (iii) **Transição pública** e a (iv) **Transição discreta**. Na CGD terá que ser identificado o tipo de transição pretendida.

A transição dos processos de *software* baseados modelos Tradicionais para processos de *software* baseados em modelos *Agile* implica um conjunto de **adaptações aos processos existentes** (Conboy *et al.*, 2010). Serão identificadas um conjunto das adaptações a serem realizadas ao processo atual da CGD.

Durante a adaptação do processo atual existem um conjunto de fatores de sucesso que influenciam positivamente a realização da transição com sucesso (Bohem and Turner, 2005; Nerur *et al*., 2005; Begel and Nagappan, 2007; Conboy *et al.*, 2010; Cohn, 2010; Lalsing *et al.*, 2012). A investigação incide sobre o conjunto de fatores de sucesso centrados nas pessoas durante a transição do processo baseado no modelo *Waterfall* para *Scrum* e nas ações que devem ser realizadas para que a transição ocorra com sucesso. Esses fatores são:

1. Motivação elevada/Aceitação do novo modelo de desenvolvimento;
2. Aprendizagem e compreensão dos valores e os princípios *Agile*;
3. Aumento generalizado das competências de desenvolvimento;
4. Aumento das competências sociais (facilitadoras da comunicação);
5. Aumento do conhecimento do negócio (facilitador da comunicação com cliente);
6. Trabalho em equipa, coerente e auto-organizado;
7. Envolvimento constante do cliente;
8. Tomada de decisão descentralizada.

# 5 MÉTODOS E TÉCNICAS DE RECOLHA E ANÁLISE DE DADOS

A abordagem metodológica deste trabalho utiliza a análise quantitativa e qualitativa. Como métodos de recolha de informação, serão utilizados a análise documental e entrevistas individuais, com o objetivo de realizar o levantamento processo atual de desenvolvimento de *software* da CGD identificando as suas ineficiências e desvantagens. Para a identificação dos fatores críticos de sucesso, também serão utilizados os mesmos métodos. Com base nesta abordagem, pretendem-se cumprir os seguintes objetivos:

1. Realizar um diagnóstico ao atual processo de desenvolvimento de *software* da CGD
    a. Levantar o atual processo de desenvolvimento de *software* da CGD;
    b. Identificar as semelhanças com os modelos teóricos de desenvolvimento de *software*;
    c. Identificar os principais problemas do processo de desenvolvimento de *software*;
    d. Analisar a experiência da CGD com modelo *Scrum*;
    e. Identificar o estado atual dos fatores de sucesso relativos às pessoas na adoção do *Scrum*.
2. Identificar as adaptações necessárias para adoção do modelo *Scrum* ao processo de desenvolvimento de *software* da CGD.
    a. Identificar as principais alterações ao processo;
    b. Desenhar uma proposta de medidas de intervenção influenciadoras do sucesso da adoção do *Scrum* na organização.
3. Definir um processo de implementação
    a. Elaborar plano com orientações, ações concretas, fases de implementação e respetivo orçamento;
    b. Elaborar cronograma de ações;
    c. Identificar os critérios de avaliação ao processo de implementação;
    d. Apresentar o valor acrescentado da implementação.

Com o objetivo de cumprir os objetivos anteriormente referidos, numa primeira fase, para a realização do diagnóstico ao atual processo de desenvolvimento de *software* da CGD será realizada uma recolha de informação com base na documentação recolhida e nas entrevistas realizadas (Anexo 1) a colaboradores que desempenham distintos papéis no atual processo de desenvolvimento. Após a recolha, esta informação será comparada com os modelos teóricos de

desenvolvimento de *software*, sendo identificado o estado atual dos fatores que influenciam o sucesso da adoção de modelos *Agile* nas organizações.

Numa segunda fase, com o objetivo de identificar as adaptações necessárias para adoção do modelo *Scrum* ao processo de desenvolvimento de *software* da CGD, de acordo com a informação recolhida na primeira fase e com base na revisão de literatura sobre os fatores de sucesso das pessoas na transição de modelos Tradicionais para modelos *Agile*, será desenhada uma proposta de intervenção para a transição do processo atual para o processo futuro, baseado no *Scrum*.

Para finalizar, será definido um processo de implementação, no qual deverão constar um plano com orientações, ações concretas, fases de implementação, respetivo orçamento. Será também disponibilizado um cronograma de ações e respetivos responsáveis. Relativamente à avaliação da implementação, serão disponibilizados os respetivos critérios, assim como o valor acrescentado para a organização.

.

# 6 A CAIXA GERAL DE DEPÓSITOS

A Caixa Geral de Depósitos, SA (CGD) é uma organização que está integrada no Grupo Caixa Geral de Depósitos (Grupo CGD), sendo o maior banco português a operar no setor financeiro. Foi criada pela Carta de Lei de 10 de abril de 1876, no reinado de D. Luís, sendo na altura Ministro da Fazenda Serpa Pimentel e presidente do 34º Governo Constitucional Fontes Pereira de Melo. Atualmente, a CGD está presente de forma integrada em todos os quadrantes do negócio bancário, nomeadamente: Banca de Investimento, Corretagem e Capital de Risco, Imobiliário, Seguros, Gestão de Ativos, Crédito Especializado, Comércio Eletrónico e Atividades Culturais.

A missão do Grupo CGD é procurar consolidar-se como um Grupo estruturante do sistema financeiro Português, distinto pela relevância e responsabilidade fortes na sua contribuição para o desenvolvimento económico, o reforço da competitividade, capacidade de inovação e internacionalização das empresas portuguesas, e estabilidade e solidez do sistema financeiro nacional.

Enquanto líder do mercado, a procura de uma evolução equilibrada entre rentabilidade, crescimento e solidez financeira, sempre no quadro de uma gestão prudente dos riscos.

O Plano Estratégico do Grupo CGD para o triénio 2011-2013 está estruturado em dois desafios chave:

1. Proteger e reforçar a saúde financeira (Solvência, Liquidez e Rendibilidade) do Grupo CGD, em resposta às necessidades geradas pelo novo enquadramento económico e do sector financeiro.
2. Transformar a CGD, focalizando a atividade no negócio bancário, de forma a assegurar a sustentabilidade e competitividade do Grupo a nível organizativo e de modelo de negócio.

A prossecução destes dois desafios está alicerçada num conjunto de 9 diretrizes, com diferentes horizontes temporais de impacto:

1. Assegurar a sustentabilidade do Funding;
2. Proteger e dinamizar a geração de receita;
3. Otimizar a eficiência em custos e eficácia da Organização;
4. Otimizar e desenvolver processos chave em acompanhamento e recuperação de crédito e em gestão do negócio imobiliário;
5. Dinamizar o crescimento rentável e sustentável da Área Internacional;

6. Otimizar a proposta de Valor em segmentos chave, em particular as Empresas e os Particulares *Affluent* e Não Residentes;
7. Otimizar o modelo e infraestrutura de abordagem comercial;
8. Promover a racionalização e excelência de processos;
9. Otimizar Políticas e Gestão de Recursos Humanos.

Os sistemas de informação da CGD são da responsabilidade da empresa do grupo, Sogrupo - Sistemas de Informação, ACE (SSI).

**O Sogrupo – Sistemas de Informação, ACE**

O Sogrupo – Sistemas de Informação, ACE (SSI) é um agrupamento complementar de empresas do Grupo CGD, sem capital próprio, composto por entidades denominadas Agrupadas. Foi constituído a 5 de Junho do ano 2000, tendo iniciado a sua atividade de prestação de serviços a 1 de Janeiro de 2001. O seu principal objetivo é a prestação, a cada uma das Agrupadas e na medida da respetiva solicitação, de serviços de conceção, desenvolvimento e exploração de soluções no domínio dos sistemas de informação, bem como a prestação de serviços de consultoria, formação e outros, de carácter técnico, associados a processos de negócio. O Sogrupo – Sistemas de Informação, ACE, é uma unidade de serviços partilhados responsável pelos sistemas de informação do Grupo CGD.

# 7 ANÁLISE DE INFORMAÇÃO E CONCLUSÕES

## 7.1 Diagnóstico situacional

O diagnóstico situacional realizado no âmbito deste trabalho de investigação tem como objetivo apresentar a situação atual da CGD no que respeita aos seguintes temas: (i) Atual processo de desenvolvimento de *software;* (ii) Utilização do *Scrum* no contexto atual; (iii) Avaliação aos fatores que influenciam sucesso de adoção do *Scrum*, relativos às pessoas. A recolha da informação necessária para aprofundar de cada um dos temas, centra-se num conjunto de entrevistas realizadas a um conjunto de colaboradores da empresa responsável pelos sistemas de informação da CGD, Sogrupo - Sistemas de Informação (SSI), participantes ativos nas equipas de desenvolvimento de *software* (Anexo I). Para além das entrevistas realizadas foi recolhida documentação que permite um melhor entendimento do atual processo de *software* da CGD, nomeadamente: (i) Metodologia de desenvolvimento de projetos do Sogrupo Sistemas de Informação (SSI), na qual documenta todo o processo de *software* da CGD, (ii) Matriz de entregáveis (Anexo II), na qual define que documentação deverá ser realizada durante o processo de desenvolvimento e (iii) Matriz de responsabilidades (Anexo III), que indica o papel de cada interveniente no processo de desenvolvimento.

### 7.1.1 *Atual processo de desenvolvimento de software*

O atual processo de desenvolvimento de *software* da CGD, apresentado no documento que define o seu modelo de desenvolvimento de *software* (Anexo I), é composto por sete fases sequenciais, apresentadas na Figura 5, que se relacionam entre si com o objetivo de entregar um produto de *software*: (i) Análise do Pedido, (ii) Análise Funcional, (iii) Desenho Técnico, (iv) Desenvolvimento, (v) Testes de Integração, (vi) Testes de Aceitação e Formação e (vii) Entrada em Produção.


**Figura 5 – Fases do atual processo de desenvolvimento de *software* da CGD**

Fonte: Documento metodologia de desenvolvimento de projetos do SSI (2011).

A Tabela 12, apresentada de seguida, pretende descrever cada uma das sete fases sequenciais do processo de desenvolvimento de *software*.

**Tabela 12 - Fases do processo de desenvolvimento de *software* da CGD**

Fonte: Documento metodologia de desenvolvimento de projetos do SSI (2011).

| Fase | Descrição |
|---|---|
| Análise do Pedido | A fase de Análise de Pedido consiste na identificação do pedido do cliente em termos de âmbito do projeto, objetivos e necessidades do negócio. Estas necessidades serão traduzidas em requisitos do cliente, numa definição de solução e numa estimativa dos custos.<br>A recolha dos requisitos do cliente é feita pela equipa da Gestão da Procura, a qual tem a responsabilidade de gerir a relação entre o cliente e os sistemas de informação. Esta etapa inclui reuniões entre o cliente, a equipa de desenvolvimento e a gestão da procura.<br>As atividades a realizar nesta fase são: (i) Elaborar Pedido; (ii) Analisar Pedido; (iii) Detalhar Requisitos; (iii) Elaborar Proposta de Solução e Orçamento; (iv) Validar e Completar Proposta; (v) Decidir e Priorizar Projetos. |
| Análise Funcional | O principal objetivo da fase de Análise Funcional é aprofundar os requisitos do cliente, transformando-os em desenhos funcionais do produto de *software* a ser desenvolvido pela equipa de desenvolvimento. Para além da equipa de desenvolvimento e do cliente, nesta fase têm intervenção, se necessário, a equipa responsável pela infraestrutura de suporte ao produto de *software*, a equipa de testes de aceitação, e a equipa responsável pela formação aos utilizadores.<br>As atividades realizadas nesta fase são as seguintes: (i) Efetuar Análise Funcional; (ii) Definir Critérios de Aceitação dos Entregáveis; (iii) Validar Análise Funcional; (iv) Elaborar Manual de Procedimentos; (v) Aprovar Análise Funcional. |
| Desenho Técnico | Após a aprovação da análise funcional pelo cliente e gestor de projeto/unidade, integrados na equipa de desenvolvimento, dá-se início à fase de Desenho Técnico.<br>É nesta fase que se procede à preparação da documentação de Desenho Técnico dividida nas seguintes atividades: (i) Efetuar Desenho Técnico; (ii) Definir Critérios de Aceitação dos |

| Fase | Descrição |
|---|---|
| | Entregáveis; (iii) Validar Desenho Técnico; (iv) Aprovar Desenho Técnico.<br>Nesta fase, para além da equipa de desenvolvimento e do gestor de projeto ou gestor da unidade, intervém também a equipa responsável pela infraestrutura de suporte ao produto de *software*. |
| Desenvolvimento | Após execução da fase de Desenho Técnico, dá-se início à fase de Desenvolvimento da exclusiva responsabilidade da equipa de desenvolvimento, na qual se inclui o gestor de projeto/unidade.<br>Nesta fase são realizadas quatro Atividades: (i) Efetuar Desenvolvimento; (ii) Definir Critérios de Aceitação dos Entregáveis; (iii) Validar Desenvolvimento; (iv) Aprovar Desenvolvimento. |
| Testes de Integração | Concluída a fase de Desenvolvimento, segue-se a fase de Testes de Integração na qual estão definidas quatro atividades: (i) Efetuar Testes de Integração; (ii) Definir Critérios de Aceitação dos Entregáveis; (iii) Validar Testes de Integração; (iv) Aprovar Testes de Integração. |
| Testes de Aceitação e Formação | Após a aprovação dos Testes de Integração segue-se fase de Testes de Aceitação e Formação. Esta fase pode contar com o envolvimento da equipa de gestão de níveis de serviço, equipa de testes de aceitação, equipa responsável pela formação dos utilizadores, o cliente, e o gestor de projeto/unidade, sendo repartida nas seguintes atividades: (i) Preparar Testes de Aceitação; (ii) Definir Níveis de Serviço; (iii) Efetuar Testes de Aceitação; (iv) Aprovar Testes de Aceitação; (v) Preparar Formação e Treino; (vi) Efetuar Formação e Treino. |
| Entrada em Produção | Após a conclusão da fase de Testes de Aceitação e Formação, segue-se a fase de Entrada em Produção.<br>Esta fase está dividida nas seguintes atividades: (i) Preparar Entrada em Produção; (ii) Efetuar Entrada em Produção; (iii) Garantir Suporte Pós-Produção; (iv) Aceitar Aplicação/Sistema. |

| Fase | Descrição |
|---|---|
| | Conta com o envolvimento da equipa de desenvolvimento, equipa responsável pela infraestrutura de suporte à aplicação/sistema, equipas SSI envolvidas na disponibilização do produto de *software*, cliente, e gestor da procura. |

Com base nas entrevistas realizadas e na Matriz de responsabilidades (Anexo III), que indica qual a responsabilidade de cada interveniente do processo de desenvolvimento de *software*, identificaram-se os seguintes intervenientes do processo de desenvolvimento de *software* da CGD: Diretor de Projeto, *Sponsor*, Responsável do Pedido/ Gestor de projeto, Equipa de Projeto (Responsável de equipa (*Team Leader*), Programadores, Analistas Funcionais, Responsáveis Técnicos), Gestor da Procura, Equipa de Planeamento e Infraestruturas, Equipa Testes de Aceitação, Equipa de Gestão de Níveis de Serviço, Equipa de Operação e Arquitetura Central, Direção de Arquitetura Estratégia e Tecnologia, Equipa de Qualidade de Melhoria Contínua, Cliente, Equipa de Formação, Arquitetura, Direção de Consultoria e Organização e o Cliente. Dos intervenientes identificados destacam-se os papéis apresentados na Tabela 13, conforme referido nas entrevistas, sendo os que maior intervenção têm no dia-a-dia do processo de desenvolvimento.

**Tabela 13 - Papéis do processo de Desenvolvimento de *software* da CGD**

Fonte: Elaboração Própria

| Papel | Descrição |
|---|---|
| Responsável pelo Pedido / Gestor de projeto | Responsável pela entrega do produto de *software* solicitado pelo cliente. |
| Responsável de Equipa | Responsável pela equipa de desenvolvimento, constituída por programadores, analistas funcionais e responsáveis técnicos, conforme as necessidades e características do produto de *software* a desenvolver. |
| Programador | Responsável pela realização do código necessário para a realização do produto de *software*. |
| Analista Funcional | Responsável pela análise funcional do produto de *software* com a qual se pretende transformar os requisitos do cliente previamente identificados num conjunto de documentos |

| Papel | Descrição |
|---|---|
| | funcionais os quais permitem fornecer à equipa de desenvolvimento o comportamento funcional do produto de *software*. |
| Responsável Técnico | Responsável pela arquitetura do sistema e pelo desenho técnico necessário à realização do produto de *software*. |

Comparando o atual processo de desenvolvimento de *software* da CGD, com o modelo Tradicional *Waterfall* publicado por Wiston W. Royce (1970) e apresentado neste trabalho de investigação no ponto 3.1.1.1, encontram-se enormes semelhanças. Tal como o modelo Tradicional *Waterfall*, trata-se de um processo de desenvolvimento de *software* baseado num plano (*plan-driven*) no qual todas as atividades do processo de desenvolvimento de *software* são planeadas e agendadas antes do processo se iniciar. Todas estas fases pressupõe um conjunto extenso de documentação, conforme se pode constatar na Matriz de Entregáveis (Anexo II), embora aligeirado pela não obrigatoriedade de realização de alguns dos documentos, fruto da maturidade do modelo ao longo dos anos. A Tabela 14 pretende resumir o estado do atual processo de desenvolvimento de *software* de acordo com as variáveis entre os modelos Tradicionais e os modelos *Agile* identificadas por Conboy *et al.* (2010).

**Tabela 14 - Atual processo CGD segundo variáveis de *Conboy***

Fonte: Elaboração própria.

| Variáveis | Atual Processo CGD |
|---|---|
| **Pressupostos** | Sistemas totalmente especificados, preditivos e construídos de acordo com um plano meticuloso e extenso. |
| **Controlo** | Centrados no processo. |
| **Estilo de gestão** | Controlo e comando. |
| **Gestão do conhecimento** | Embora formalmente explicito acaba por centrar-se nas pessoas. |
| **Atribuição de papéis** | Individuais – favorece a especialização. |
| **Comunicação** | Formal com o cliente, recurso a muita documentação.<br>Informal entre equipas e entre elementos da equipa. |
| **Papel do cliente** | Importante, sendo maioritariamente envolvido no início e no fim do processo de desenvolvimento. |

| | |
|---|---|
| **Ciclo do projeto** | Orientado a tarefas ou atividades. |
| **Modelo desenvolvimento** | *Waterfall.* |
| **Organização/Estrutura** | Mecânica (Burocrática com elevada formalidade). |
| **Tecnologia** | Várias tecnologias utilizadas. |
| **Localização equipa** | Equipas distribuídas. |
| **Tamanho equipa** | Bastante variável. |
| **Aprendizagem contínua** | Existe e é encorajada apenas aos recursos internos à CGD. |
| **Cultura de gestão** | Controlo e comando. |
| **Participação da equipa** | Não obrigatória. |
| **Planeamento projeto** | Preditivo e pouco adaptativo. |
| **Mecanismos de *feedback*** | Não obtido facilmente, pois o cliente participa mais no inicio e no fim do desenvolvimento. |
| **Documentação** | Substancial e necessária para finalizar as várias fases do processo de desenvolvimento. |

Nas entrevistas realizadas, a experiência diária de utilização do processo de desenvolvimento de *software* da CGD permitiu às pessoas identificarem os seguintes de aspetos negativos:

- Especificação detalhada no início do processo;
- Longos ciclos de desenvolvimento;
- Muita documentação;
- Elevado custo de mudança;
- Entrega tardia de valor para o cliente;
- Fraco envolvimento do cliente;
- Inexistência de uma visão global na constituição de requisitos;
- Falta de autoridade e capacidade crítica das equipas de desenvolvimento;
- Testes somente no final do processo.

A maioria destes aspetos fazem parte das desvantagens identificadas por Petersen (Petersen *et al*., 2009) relativamente aos processos baseados em modelos *Waterfall*.

### 7.1.2 A utilização do *Scrum* no contexto atual

De acordo com as entrevistas realizadas, no ano de 2010, a CGD começa a utilizar o modelo *Scrum* numa das suas equipas de desenvolvimento de *software*, nomeadamente a equipa que

desenvolveu o Processo de Crédito Empresas. No seguimento da experiência de desenvolvimento obtida, incluiu-se, no plano de formação anual, uma formação interna de iniciação ao modelo *Scrum*, tendo sido até ao momento formados mais de 100 dos 650 colaboradores participantes do processo de desenvolvimento de *software*. A CGD atualmente tem aproximadamente 80 pessoas a utilizar o modelo *Scrum* no seu processo de desenvolvimento de *software*. Durante este tempo, a organização permitiu a utilização deste processo com a condição de que as equipas cumprissem com as regras do atual processo de desenvolvimento de *software* da CGD. Na prática, de acordo com um responsável por uma destas equipas, o processo seguido por estes pioneiros é um processo semelhante ao modelo *Water-Scrum-Fall* descrito por West (2011), adaptando o processo de desenvolvimento de *software* atual às práticas seguidas pelo modelo *Scrum*, integrando algumas práticas do *Scrum* no modelo atual.

Um dos programadores de uma equipa das equipas que utiliza *Scrum*, referiu a dificuldade de integração entre o *software* produzido pela equipa de *Scrum* e as restantes equipas que não funcionam em *Scrum*. As dificuldades também ocorrem com as entregas frequentes, as quais são muitas vezes difíceis de compatibilizar com as poucas e apenas entregas finais do atual processo de desenvolvimento de *software* da CGD. Este cenário é semelhante ao descrito por Sliger (2006), o *Waterfall-in-tandem*, no qual duas equipas que contribuem para o desenvolvimento de um produto de *software*, funcionando uma com um processo *Waterfall* e a outra com um processo *Scrum*.

### 7.1.3 Avaliação aos fatores que influenciam o sucesso da adoção do *Scrum*

Para cada um dos fatores de sucesso da adoção do *Scrum* relativos às pessoas, identificados no ponto 3.4.3, foi realizado um levantamento do seu estado atual, baseado nas entrevistas realizadas aos colaboradores da CGD, participantes do atual processo de desenvolvimento de *software* da CGD. Pretende-se saber, desta forma, qual o ponto de partida para a transição entre o atual processo e o futuro processo de desenvolvimento de *software.*

**#1 Motivação elevada/Aceitação do novo modelo de desenvolvimento.** De acordo com as entrevistas realizadas, foram identificados os seguintes grupos de pessoas no que respeita à motivação/aceitação do novo modelo de desenvolvimento de *software*:

- Pioneiros – grupo de pessoas que por sua iniciativa iniciaram a utilização do modelo *Scrum* nas suas equipas, motivados pelos seguintes fatores: (i) Sucesso das equipas que utilizam o modelo *Scrum* na CGD, (ii) Dificuldades sentidas no atual processo de

desenvolvimento, (iii) Participação na formação sobre iniciação ao *Scrum*. Representam aproximadamente 15% dos 650 colaboradores associados ao desenvolvimento de *software* na CGD.

- Céticos – grupo de pessoas que não acreditam na mudança. No passado já ocorreram outros processos de mudança dentro da organização que não tiveram o impacto esperado, criando um sentimento de desilusão e desmotivação. Estas experiências negativas influenciaram as pessoas ao longo do tempo, para as quais, deverá existir um olhar atento e um conjunto de medidas que as envolva no processo de mudança;
- Conservadores – grupo de pessoas que prefere continuar com o modelo atual. Estas pessoas têm medo da mudança, preferem a tradição e a prática corrente. O seu conservadorismo acentua-se pelo desconhecimento do que vai ser adotado e em que medida serão afetados.

Os Céticos e os Conservadores deverão ser alvo de especial atenção durante todo o processo de mudança. Para alguns dos entrevistados estes dois grupos representam a maioria das pessoas envolvidas no processo de desenvolvimento da CGD.

**#2 Aprendizagem e compreensão dos valores e dos princípios *Agile*.** A aprendizagem e compreensão dos valores e dos princípios *Agile*, de acordo com as entrevistas realizadas, tem sido realizada com base nas seguintes iniciativas: (i) Formação interna de iniciação ao *Scrum*, (ii) Formação interna de iniciação ao *Lean*, (iii) *Coaching Lean* realizado nas unidades. Com o início da utilização do modelo *Agile*, no ano de 2010, introduziu-se no plano de formação interna a formação de iniciação ao *Scrum*, na qual são passados os valores e princípios *Agile*. No passado ano de 2007, de acordo com as respostas às entrevistas, o SSI iniciou a adoção do modelo de gestão *Lean*, com o objetivo de simplificar a organização como um todo, no que respeita aos sistemas de informação. O modelo de gestão *Lean* adotado, o qual não deve ser confundido com o modelo *Agile* de desenvolvimento de *software*, é um modelo de gestão que procura melhorar a eficácia e a eficiência dos processos da organização minimizando o que não acrescenta valor e que não é necessário para satisfazer os requisitos do cliente, ou seja, desperdícios, variabilidade e inflexibilidade. Muitos dos princípios do modelo de gestão *Lean* são idênticos aos valores e princípios *Agile*, sendo transmitidos nas formações administradas aos colaboradores do SSI. Ainda no âmbito do *Lean*, foram tornados obrigatórios os "Quadros Brancos" em cada unidade, com o objetivo tornar visíveis as tarefas diárias de cada colaborador. À volta dos mesmos é realizada diariamente uma Reunião diária, semelhante à Reunião diária do *Scrum*, no qual cada colaborador deverá dizer o que fez ontem, o que tem para fazer hoje e

quais os seus impedimentos. Para ajudar à adoção do *Lean* nas unidades do SSI, foi criado um modelo de *coaching*, o qual atribui a cada unidade um *coach Lean*, com a responsabilidade de ajudar as equipas a seguirem as melhores práticas.

**#3 Aumento generalizado das competências de desenvolvimento**. A CGD, de acordo com as entrevistas realizadas, não se foca no aumento das competências generalizadas de desenvolvimento dos seus colaboradores. Apesar da existência de um plano de formação focado na especialização dos colaboradores internos, este não abrange os colaboradores externos, que representam a maioria das pessoas que participam no processo de desenvolvimento. A CGD, por regra, não intervém diretamente nos planos de formação dos seus colaboradores externos, deixando para respetivas empresas a decisão da formação a disponibilizar aos seus colaboradores.

**#4 Aumento das competências sociais (facilitadoras da comunicação).** Com a introdução dos "Quadros Brancos" no âmbito da adoção do modelo de gestão *Lean* ao SSI, ajudada pela presença dos *coaches* em cada unidade, a comunicação entre os elementos de equipa tornou-se obrigatoriamente diária. De acordo com as entrevistas realizadas, este modelo de "Quadro Branco" encontra-se mais orientado para as unidades e não tanto para as equipas de projeto. De uma forma generalizada, referido também nas entrevistas, a comunicação dentro do SSI é formal com o cliente, com a utilização preferencial do *mail* e/ou reuniões com elevado número de participantes. No que diz respeito à comunicação entre e dentro das equipas de desenvolvimento, normalmente é utilizada a comunicação informal, via telefone, *mail* ou cara a cara, dispensando quaisquer formalismos.

**#5 Aumento do conhecimento do negócio (facilitador da comunicação com cliente).** Na CGD existe uma preocupação constante em passar conhecimento do negócio bancário através de *e-learning*, de acordo com um dos entrevistados. Existem um conjunto de formações anuais, obrigatórias que cada colaborador tem a obrigação de realizar. No entanto estas formações não estão alinhadas com os projetos que se estão a realizar. Um dos problemas apresentado nas entrevistas aborda o facto dos colaboradores externos não encontrarem abrangidos por este tipo de formação, sendo que muitos deles chegam a ter mais de 10 anos de casa e representam a maioria dos participantes no processo de desenvolvimento de *software.*

**#6 Trabalho em equipa, coerente e auto-organizado.** De acordo com as entrevistas realizadas, as equipas de desenvolvimento dependem do gestor de projeto e/ou responsável da unidade na qual o projeto reside para se organizarem. Relativamente às equipas que trabalham com o modelo *Scrum*, embora tenham um maior nível de maturidade no que respeita ao trabalho

conjunto e auto-organizado, ainda não se podem considerar equipas auto-organizadas, carecendo da presença de um responsável que os organize.

**#7 Envolvimento constante do cliente.** O envolvimento do cliente é maior no início para o levantamento de requisitos e no fim para a realização de testes de aceitação. Este facto pode ser observado no documento de Metodologia de desenvolvimento de projetos do Sogrupo Sistemas de Informação (SSI), na qual documenta todo o processo de *software* da CGD, assim como nas respostas às entrevistas. A viver uma realidade diferente, os entrevistados pertencentes a equipas de *Scrum*, referem a disponibilidade do cliente, assim como envolvimento no dia-a-dia do processo de desenvolvimento.

**#8 Tomada de decisão descentralizada.** De acordo com um dos entrevistados, existe uma falta de autoridade e de capacidade crítica das equipas de desenvolvimento. Globalmente e provavelmente pelo modelo de contratação de recursos externos, conforme referido em entrevista, a tomada de decisão ainda está muito dependente das chefias diretas, o que torna as equipas dependentes.

Finalizado o levantamento do processo de desenvolvimento de *software* da CGD, da utilização do *Scrum* e da avaliação aos fatores que influenciam o sucesso da adoção do *Scrum*, obtém-se neste trabalho de investigação o ponto de partida para a transição do atual processo para um futuro processo baseado no modelo *Scrum*.

## 7.2 Solução a adotar

Tendo por base o diagnóstico situacional, esta secção tem como objetivo a apresentação da versão base do futuro processo de desenvolvimento de *software* para a CGD e um conjunto de medidas que visam potenciar o sucesso da adoção do modelo *Scrum* ao processo de desenvolvimento da CGD, transitando do atual processo Tradicional para um processo *Agile* baseado no modelo *Scrum*. A versão base do futuro processo de desenvolvimento de *software* deverá ser trabalhada e aprofundada pelas pessoas envolvidas ao longo do projeto de implementação, com vista à obtenção de um processo mais adequado à realidade e ao *buy-in* das pessoas afetadas.

### 7.2.1 *Futuro processo de desenvolvimento*

Por forma a cumprir o objetivo deste trabalho de investigação, baseado nas entrevistas realizadas e no diagnóstico situacional, o futuro processo de desenvolvimento deverá cumprir as seguintes linhas de orientação:

- **Basear-se no modelo *Scrum*.** É o objetivo deste trabalho de investigação e da CGD transitar do atual processo de desenvolvimento para um processo de desenvolvimento baseado no modelo *Scrum*. Para além do processo se basear nas regras do modelo, deverá cumprir os valores e princípios *Agile*, nomeadamente: (i) Indivíduos e interações em vez de processos e ferramentas; (ii) *Software* a funcionar em vez de documentação; (iii) Colaboração do cliente em vez de negociação de contratos; (iv) Resposta à mudança em vez de seguir um plano. A utilização do modelo como referência para o novo processo de desenvolvimento permitirá eliminar os aspetos negativos do atual processo de desenvolvimento referidos nas entrevistas e apresentados no diagnóstico situacional deste trabalho de investigação: (i) Especificação detalhada no início do processo; (ii) Longos ciclos de desenvolvimento; (iii) Muita documentação; (iv) Elevado custo de mudança; (v) Entrega tardia de valor para o cliente; (vi) Fraco envolvimento do cliente; (vii) Inexistência de uma visão global na constituição de requisitos; (viii) Falta de autoridade e capacidade crítica das equipas de desenvolvimento; (ix) Testes somente no final do processo.
- **Manter uma fase inicial de análise de pedido.** De acordo com Sliger (Sliger, 2006), as regras de governação das organizações requerem que sejam definidos os requisitos e os planos antes de se iniciar o trabalho de desenvolvimento. Em muitas organizações, em particular na CGD, estes planos formam o contrato entre o negócio e os sistemas de informação, o qual define a direção do projeto, a sua *timeline* e o seu orçamento. Sliger (2006) recomenda que se siga o conselho de Alistair Cockburn (2000) e seja produzida a documentação estritamente necessária e suficiente para esta fase, em vez de se tentar especificar e documentar detalhadamente todo o produto de *software*.
- **Ter em conta as várias realidades da organização.** O futuro processo de desenvolvimento de *software* deverá ser universal à organização, para se adaptar às diversas equipas de desenvolvimento que trabalham em tecnologias distintas, para diferentes clientes. O contributo de cada um deles será determinante ao longo do processo de implementação para se ajustar o processo à realidade das equipas envolvidas.
- **Ter em conta a experiência existente com *Scrum*.** A CGD tem a oportunidade de criar um processo de desenvolvimento de *software* com base na experiência diária das equipas que desde do ano de 2010 utilizam o modelo *Scrum* no seu processo de desenvolvimento.

A versão base do processo de desenvolvimento de *software* da CGD apresentada neste trabalho de investigação é composta por: (i) Fases do processo de desenvolvimento; (ii) Papéis e Responsabilidades, apresentados de seguida. A documentação necessária é referida ao longo das fases do processo de desenvolvimento.

#### 7.2.1.1 Fases do processo de desenvolvimento

Tendo por base as linhas orientadoras definidas no ponto anterior, as fases do novo processo de desenvolvimento da CGD baseado no modelo *Scrum* (ver Figura 6 abaixo) são as seguintes: Análise do Pedido; Desenvolvimento; Conclusão do Pedido.


**Figura 6 – Fases do novo processo de desenvolvimento da CGD**

Fonte: Elaboração própria.

**Análise do Pedido.** A existência de uma fase de Análise do Pedido é necessária para a maioria das organizações, em particular para a CGD. Esta fase baseia-se na atual fase de Análise do Pedido, diferenciando-se na redução do detalhe da especificação e na documentação a ser realizada. O objetivo desta fase é traduzir as necessidades do cliente em três componentes: (i) Definição geral da solução a adotar, (ii) *Backlog* do produto de alto nível e (iii) Estimativa de alto nível do tempo e custos envolvidos. Após as necessárias sessões de trabalho e esclarecimento que podem envolver o cliente, a equipa de desenvolvimento, *Product owner*, gestor de projeto e a gestão da procura, os três componentes deverão ser incluídos no já existente documento interno, Informação de Projeto, sujeito à aprovação da hierarquia para que o projeto possa prosseguir e passar à fase de Desenvolvimento.

**Desenvolvimento.** A fase de Desenvolvimento inicia-se com a aprovação do documento interno, Informação do Projeto, pela hierarquia. É baseada no modelo *Agile*, *Scrum*, e tem como objetivo principal a entrega incremental do produto de *software* ao cliente, não retendo o valor até ao final do processo. As atividades desta fase de Desenvolvimento, apresentadas na Figura 7, são idênticas às atividades realizadas pelas equipas que utilizam *Scrum* na CGD, não tendo sido encontradas evidências de necessidade de mudança ao longo das entrevistas e do

levantamento realizado. Dividem-se em: (i) atividades contínuas e (ii) atividades recorrentes. As atividades contínuas são realizadas continuamente ao longo de toda a fase de desenvolvimento, nomeadamente as atividades de Gestão do *Backlog* de produto e de Reunião diária. As atividades recorrentes com o objetivo de desenvolver o incremento de produto de *software* são as seguintes: (i) Planeamento do *Sprint*; (ii) Desenvolvimento do produto de *software*; (iii) Revisão do Desenvolvimento; (iv) Retrospetiva do Desenvolvimento. Por fim, sobram as atividades recorrentes que têm como objetivo a aceitação e disponibilização do produto de *software*: (v) Aceitação do Incremento de Produto de *Software*; (vi) Entrega do produto de *software* aos utilizadores finais.

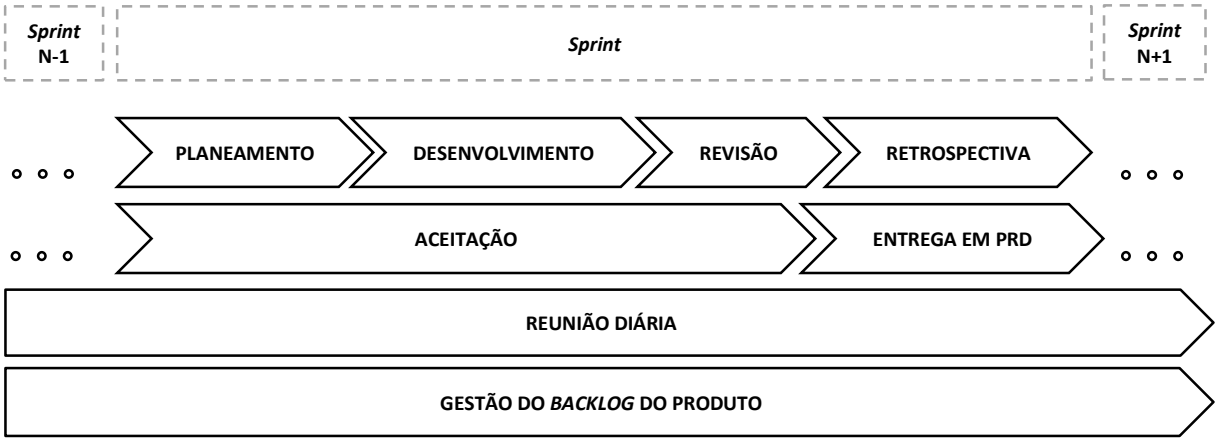


**Figura 7 - Atividades da fase de desenvolvimento do novo processo de *software***

Fonte: Elaboração própria.

As atividades da fase de desenvolvimento são explicadas na Tabela 15, abaixo.

**Tabela 15 - Descrição das atividades da fase de desenvolvimento**

Fonte: Elaboração própria.

| Atividade | Descrição |
|---|---|
| Gestão do *Backlog* de produto | A gestão do *Backlog* do produto de *software*, de acordo com o modelo *Scrum*, é uma atividade da responsabilidade do *Product Owner*. Este deverá continuamente alimentar o *Backlog* de produto com os requisitos dos *stakeholders* – qualquer pessoa ou organização que tenha interesse ou seja afetada pelo projeto. O *Backlog* de produto deverá ser regularmente priorizado de acordo com as |

| Atividade | Descrição |
|---|---|
| | alterações ou novos requisitos dos *stakeholders*, baseando-se sempre no valor para o negócio e no custo de desenvolvimento de cada *story*. As *stories* deverão sempre conter sempre os critérios de aceitação. Estes critérios de aceitação permitem saber se a *story* foi disponibilizada ao utilizador final de acordo com os requisitos. O detalhe de cada *story* deverá ser maior para as *stories* de topo, pois serão as primeiras a serem desenvolvidas. |
| Reunião diária | Como definido no modelo *Scrum*, a reunião diária é realizada diariamente durante a atividade de desenvolvimento. Esta reunião tem como objetivo a comunicação entre os elementos da equipa e a inspeção do *Sprint*. Com a duração de 15 minutos no máximo, deverá ser realizada de pé. Cada elemento da equipa deverá responder a três questões:<br>1. O que fiz ontem?<br>2. O que vou fazer hoje?<br>3. Quais os meus impedimentos? |
| Planeamento | A atividade de planeamento do *Sprint* permite que a equipa de desenvolvimento planeie o trabalho a realizar em cada ciclo de desenvolvimento. Antes do início de cada *Sprint*, a equipa de desenvolvimento e o *Product Owner*, com base no *Backlog* de produto, chegam a acordo sobre as *stories* a desenvolver. As *stories* de topo do *Backlog* de produto deverão ser as elegíveis para serem desenvolvidas. O planeamento termina com a elaboração do *Backlog* do *Sprint*. Este artefacto contém as tarefas a realizar para a conclusão de cada uma das *stories* escolhidas para serem desenvolvidas durante o *Sprint*. |
| Desenvolvimento | Durante a atividade de desenvolvimento, coração do processo de desenvolvimento, a equipa de desenvolvimento cria uma versão incremental e potencialmente utilizável do produto de *software*. A sua duração, definida na atividade de planeamento do *Sprint*, deverá ser de uma semana a um mês. A atividade de testes integrados, |

| Atividade | Descrição |
|---|---|
| | existente no atual processo de desenvolvimento de *software*, passa a fazer parte da atividade de desenvolvimento. |
| Aceitação | Nos testes de aceitação é obtida a confirmação de que as *stories* desenvolvidas satisfazem os requisitos do cliente, cumprindo os respetivos critérios de aceitação. Os testes de aceitação devem ser realizados pelo cliente em conjunto com a equipa de desenvolvimento (Analistas Funcionais/*Testers*). |
| Entrega em Produção | Atividade na qual é disponibilizado o incremento do produto de *software* aos utilizadores finais. |
| Revisão do Desenvolvimento | A reunião de Revisão do *Sprint*, realizada no final do *Sprint*, permite que a equipa apresente ao cliente ou a qualquer parte interessada, o incremento de produto realizado. A partir do *feedback* dos presentes, adapta o *Backlog* do produto se necessário. O objetivo, com base no que foi produzido, é identificar o que poderá ser alterado nos próximos *Sprints*, de forma a otimizar o valor do produto para o cliente. Esta reunião não deve ultrapassar a duração de 4 horas para *Sprints* de um mês de duração, sendo este tempo adaptado conforme a duração do *Sprint*. |
| Retrospetiva | Trata-se de uma oportunidade para a equipa olhar para ela mesma, identificando pontos de melhoria a aplicar no próximo *Sprint*. A equipa inspeciona-se a si mesma e adapta-se. Esta reunião, tal como a anterior, não deve ultrapassar a duração de 4 horas para *Sprints* de um mês de duração, sendo este tempo adaptado conforme a duração do *Sprint*. |

**Conclusão do Pedido.** Nesta fase é concluído o processo de desenvolvimento do produto de *software*. Para que o processo possa ser formalmente dado como concluído deverão ser realizadas as seguintes atividades:

- Disponibilizar a documentação de arquitetura do sistema;
- Disponibilizar os manuais do produto de *software*;
- Realizar a passagem da informação/documentação necessária para as equipas que vão manter o produto de *software*;

- Registar e partilhar as lições aprendidas no projeto;
- Formalizar a conclusão do pedido junto dos *Stakeholders*.

#### 7.2.1.2 Papéis e Responsabilidades

Após definição das fases do processo e as respetivas atividades, importa definir os papéis e responsabilidades de cada interveniente do processo de desenvolvimento. De acordo com Mike Cohn (2010), para além da criação dos papéis de *Scrum master* e *Product owner*, deverão ser revistas as responsabilidades dos papéis existentes na organização, para que se encaixem nos papéis do *Scrum* (Schwaber and Sutherland, 2013): (i) *Scrum master*; (ii) *Product owner;* (iii) Equipa de desenvolvimento. Para cada um dos papéis identificados na análise e diagnóstico ao atual processo de desenvolvimento de *software*, foram revistas as suas responsabilidades para que se adequem ao novo processo de desenvolvimento e se encaixem nos papéis do modelo *Scrum*, conforme se pode observar na Tabela 16, em baixo. A esta revisão aos papéis existentes acrescentam-se os papéis de *Scrum master, Product Owner* e Equipa de desenvolvimento.

**Tabela 16 - Revisão dos Papéis e responsabilidades**

Fonte: Elaboração própria

| Papel | Descrição |
|---|---|
| Gestor da Procura | O Gestor da Procura tem um papel importante na organização na fase de Análise de Pedido, a qual se manterá no futuro processo de desenvolvimento de *software*. Por este motivo, deverá manter-se o papel assim como as suas responsabilidades atuais: (i) Realização do caderno de requisitos em conjunto com o Cliente na fase inicial de Análise de Pedido; (ii) Garantir a execução da Proposta de Solução, do Orçamento e da justificação do Pedido; (iii) Realização dos inquéritos intercalares e finais de satisfação do cliente. No entanto, deverá focar a sua atenção para que a fase a Análise de Pedido seja realizada com a maior celeridade possível e que os requisitos do cliente fiquem espelhados não no caderno de requisitos mas sim numa primeira versão do *Backlog* de produto, gerida pelo *Product Owner*. |

| Papel | Descrição |
|---|---|
| Gestor de Projeto ou Responsável pelo Pedido | Pela dimensão da organização e dos projetos que nela se realizam, torna-se importante manter este papel, não pertencente ao modelo *Scrum*, assim como as suas atuais responsabilidades: (i) Realização da Proposta de Solução e do Orçamento; (ii) Realização da Informação de Projeto que permite a aprovação do pedido pela hierarquia; (iii) Gestão de *stakeholders*, âmbito, tempo, custo e riscos do pedido; (iv) Registo das lições aprendidas. |
| Analista Funcional | O Analista Funcional mantém as responsabilidades que tem atualmente e dado que tem uma parte de responsabilidade idêntica à do *Product Owner*, deverá ser convertido neste papel do *Scrum*. |
| *Tester* | O papel de *Tester*, tal como o papel de Analista Funcional deverá ser convertido no papel de *Product Owner* dado que este tem a responsabilidade pela qualidade da entrega do produto de *software*. |
| Responsável Técnico | O Responsável Técnico é atualmente um especialista dentro da equipa de desenvolvimento, sendo a equipa de desenvolvimento um dos papéis do *Scrum*, sem atribuição específica de papéis aos especialistas. Como referido por Conboy (2010), é importante que estes especialistas ganhem competências no maior conjunto de áreas possível, para que possam ajudar a equipa a desenvolver o produto de *software*. Em suma, este papel deverá ser extinto, embora continuem a contribuir com o seu especializado conhecimento para o desenho técnico e desenvolvimento do produto de *software*. |
| Programador | Mantém o seu papel e responsabilidades atuais, sendo-lhe dada mais liberdade e responsabilidade para organizar o seu trabalho e o da equipa. Faz parte da equipa de desenvolvimento, papel do *Scrum*. |
| *Scrum Master* | O *Scrum master* é o responsável por assegurar a compreensão e o bom funcionamento do *Scrum* no seio da equipa. O *Scrum master* é um líder que tem que se disponibilizar para servir a equipa. Gere as interferências externas, ajudando os elementos externos à equipa a entender quais das suas interações com a mesma são úteis ou não. |
| *Product Owner* | É o responsável por maximizar o valor do produto e do trabalho da equipa de desenvolvimento. É o elemento responsável por gerir o |

| Papel | Descrição |
|---|---|
| | *Backlog* do produto transformando os requisitos dos clientes e da equipa, em funcionalidades do produto. Participa na estimativa do esforço de desenvolvimento. Realiza os testes ao produto de *software*. |
| Equipa de desenvolvimento | A equipa de desenvolvimento consiste num conjunto de profissionais que trabalham para entregar um "incremento de produto" em cada *Sprint*. As equipas de desenvolvimento devem ser auto-organizadas e compostas por elementos com conhecimentos transversais, necessários para o desenvolvimento do produto. |

Em suma, os papéis do futuro processo de desenvolvimento de *software* são: Gestor da Procura, Gestor de Projeto/Responsável Pedido, *Product Owner*, *Scrum Master* e Equipa de desenvolvimento.

Com as fases e os papéis do futuro processo de desenvolvimento, temos um processo base para dar início à transição entre o atual processo e o futuro processo. As medidas que influenciam o sucesso da adoção do *Scrum*, apresentadas no ponto de seguinte, completam a solução a adotar.

### 7.2.2 *Medidas que influenciam o sucesso da adoção do Scrum*

Pretende-se nesta secção apresentar um conjunto de ações que levem à maximização dos fatores de sucesso, relativos às pessoas, da adoção do modelo *Scrum* (ponto 3.4.3), nomeadamente:

1. Motivação elevada/Aceitação do novo modelo de desenvolvimento
2. Aprendizagem e compreensão dos valores e os princípios *Agile*;
3. Aumento generalizado das competências de desenvolvimento;
4. Aumento das competências sociais (facilitadoras da comunicação);
5. Aumento do conhecimento do negócio (facilitador da comunicação com cliente);
6. Trabalho em equipa, coerente e auto-organizado;
7. Envolvimento constante do cliente;
8. Tomada de decisão descentralizada.

O conjunto de recomendações de maximização de cada um dos fatores de sucesso da adoção do *Scrum*, fornecidas por Nerur *et al*. (2005), Bohem e Turner (2005); Mahanti (2006), Conboy

*et al.*, (2008), Cohn (2010) e Gandomani *et al.* (2014), apresentadas na Tabela 17, servem como ponto de partida para as medidas a implementar.

**Tabela 17 - Recomendações de maximização fatores sucesso da adoção *Scrum***

Fonte: Nerur *et al*. (2005), Bohem e Turner (2005); Mahanti (2006), Conboy *et al*., (2008), Cohn (2010) e Gandomani *et al*. (2014)

| # | Fator de Sucesso | Recomendações de maximização |
|---|---|---|
| 1 | Motivação elevada/Aceitação do novo modelo de desenvolvimento | • Plano de comunicação eficaz;<br>• Envolver as pessoas no processo de transição, principalmente os mais céticos;<br>• Realizar pilotos com sucesso em diferentes tipos de projetos;<br>• Alinhar os incentivos existentes com os futuros;<br>• Tentar ter sempre pessoas *"bought-in"* em cada equipa. |
| 2 | Aprendizagem e compreensão dos valores e os princípios *Agile* | • Formação contínua;<br>• *Coaching* em processos de *software Agile*. |
| 3 | Aumento generalizado das competências de desenvolvimento | • Utilizar *pair programming* e *pair rotation*. São técnicas de desenvolvimento de *software* que permitem a partilha de conhecimento e facilitam a aprendizagem;<br>• Encorajar a autoatribuição de tarefas;<br>• Reintroduzir papéis específicos;<br>• Formação. |
| 4 | Aumento das competências sociais (facilitadoras da comunicação) | • Formação em competências sociais;<br>• Utilização de documentação apropriada que facilite a comunicação; |
| 5 | Aumento do conhecimento do negócio (facilitador da comunicação com cliente) | • Formação;<br>• Recrutar colaboradores com conhecimento no negócio. |
| 6 | Trabalho em equipa, coerente e auto-organizado | • Focar a responsabilidade na equipa e não no individuo;<br>• Realizar uma boa e moderada utilização dos especialistas; |

| # | Fator de Sucesso | Recomendações de maximização |
|---|---|---|
| | | • Paralelizar tarefas da equipa;<br>• Aumentar e partilhar conhecimento constantemente;<br>• Introduzir alterações nas equipas, alterando contentores, atenuando diferenças ou alterando formas de interação entre os elementos da equipa. |
| 7 | Envolvimento constante do cliente | • Participação ativa e constante do cliente no processo de desenvolvimento. |
| 8 | Tomada de decisão descentralizada | • Construir um ambiente de partilha e aprendizagem para dar à equipa capacidade de tomada de decisão;<br>• Implementar um sistema democrático de voto;<br>• O gestor de projeto deverá desempenhar um papel de facilitador. |

Com base no diagnóstico situacional e nas recomendações descritas na Tabela 17, foram identificadas um conjunto de medidas a realizar, as quais abrangem a totalidade das recomendações identificadas, influenciando positivamente estes fatores de sucesso na adoção do nosso processo de desenvolvimento, nomeadamente:

1. Plano de comunicação eficaz;
2. Envolvimento das pessoas;
3. Plano de formação para os colaboradores internos e externos que aborde as componentes de *Scrum*, Técnicas de Desenvolvimento, competências Sociais e de Negócio;
4. *Coaching*;
5. Realização de projetos piloto

1. **Plano de comunicação eficaz**. A comunicação sobre que se pretende realizar, assim como a informação sobre o estado da mudança, eleva a motivação e a aceitação do novo modelo de desenvolvimento por parte dos colaboradores da organização, de acordo com Bohem e Turner (2005), Conboy *et al.* (2010) e Cohn (2010). Conforme as entrevistas realizadas, comparando com outras alterações processuais da CGD, é considerada uma boa prática a partilha e a comunicação para que exista um envolvimento das pessoas. A comunicação deve ter em conta o seguinte:

(i) A Administração do SSI deverá comunicar as razões da mudança a todos os colaboradores do SSI;
(ii) Os superiores hierárquicos diretos deverão comunicar como esta mudança deverá afetar cada um dos colaboradores das suas equipas;
(iii) Criar oportunidades para os pares partilharem as suas experiências.

2. **Envolvimento das pessoas.** O envolvimento das pessoas, principalmente dos mais céticos, melhora os níveis de aceitação do novo processo de desenvolvimento de *software* (Cohn, 2010). O desenho do novo processo de desenvolvimento de *software* deverá ser realizado pelas pessoas e corrigido ao longo do tempo, com base nas experiências vividas em cada projeto. Representantes dos principais intervenientes do processo de desenvolvimento, incluindo clientes, deverão ser envolvidos no desenho do novo processo e participar na implementação. O envolvimento transversal da organização é muito importante para a existência *buy-in* desta mudança. Para além deste *buy-in*, a existência de múltiplas realidades no atual processo de desenvolvimento, que devem ser escutadas e incluídas ou adaptadas ao novo processo de desenvolvimento de *software*.

3. **Plano de formação.** A formação das pessoas desempenha um papel bastante importante na adoção de processos baseados no modelo *Scrum*, potenciando um vasto conjunto de fatores de sucesso, nomeadamente: a aprendizagem dos princípios e valores *Agile*, o aumento das competências de desenvolvimento, sociais e o conhecimento do negócio. A formação a realizar deverá focar o seguinte: (i) *Agile*/*Scrum*; (ii) Técnicas de desenvolvimento; (iii) Técnicas de comunicação; (iv) Conceitos do negócio.
    (i) ***Agile*/*Scrum*.** Para a existência do conhecimento sobre o processo de desenvolvimento de *software* baseado no modelo *Scrum*, para além do conhecimento sobre as atividades que o constituem, é necessária a aprendizagem e compreensão dos valores e princípios *Agile* (Conboy *et al.*, 2010). Para que a formação sobre os valores e princípios *Agile*, seja funcional e adequada, a formação deverá ser contínua, abranger a totalidade das pessoas envolvidas no processo de desenvolvimento, ter conteúdos formativos adequados e ter uma forte componente prática (Gandomani *et al.*, 2014). Propõe-se a criação de um programa formativo, contínuo, envolvendo formações e conferências, que mantenham vivos os valores e princípios *Agile* ao longo do tempo. Este programa de formação deverá envolver todos os participantes do processo de desenvolvimento de *software* da CGD. O

objetivo inicial será a formação em *Scrum* de todos os participantes no processo de desenvolvimento de *software* da CGD;

(ii) **Técnicas de desenvolvimento.** O aumento de produtividade na adoção de processos baseados em modelos *Agile*, encontra-se associado à utilização de técnicas *Agile* de desenvolvimento de *software* (Cohn, 2010). Propõe-se um programa de formação específico na utilização de técnicas tais como: *test-driven development, refactoring, continous integration* ou *pair programming*. Este programa de formação deverá ter como público-alvo todos os participantes do processo de desenvolvimento de *software* que lidem com os aspetos técnicos do desenvolvimento e os gestores;

(iii) **Técnicas de comunicação.** As práticas *Agile* levam a um aumento da interação social que carece de capacidade de sociabilização, comunicação e apresentação (Lalsing *et al.,* 2012). Propõe-se um programa de formação contínua em competências sociais, focado na comunicação. Esta formação contínua deverá ser extensível a todos os participantes no processo de desenvolvimento de *software*;

(iv) **Conceitos de negócio.** O desenvolvimento *Agile* implica uma constante interação entre o cliente e a equipa de desenvolvimento, com base numa linguagem comum, baseada no conhecimento do negócio (Conboy *et al.*, 2010). Propõe-se a realização de sessões periódicas sobre conceitos básicos do negócio. Estas sessões, administradas por pessoas do negócio, deverão ter como objetivo a apresentação de conceitos básicos sobre o mesmo. No início de cada projeto, deverá realizar-se uma sessão formativa que permita à equipa adquirir os conhecimentos de negócio para o desenvolvimento do seu projeto.

Um dos problemas identificados no diagnóstico situacional é o facto dos colaboradores externos, contratados de num regime de *outsourcing* especializado ou de prestação de serviços, não serem abrangidos pelo plano de formação da CGD ou a CGD não ter interferência direta na sua formação. Para que o plano de formação tenha a influência necessária para o sucesso da adoção do modelo *Scrum* deverá abranger todas as pessoas envolvidas no desenvolvimento de *software,* internos e externos. Com base neste pressuposto, recomenda-se a revisão dos contratos com as empresas que fornecem recursos externos à CGD.

4. ***Coaching***. O *coaching* é fundamental para a correta adoção do novo processo de desenvolvimento de *software*, em particular na compreensão dos valores e princípios *Agile*. O objetivo é formar um conjunto de *coaches* em *Scrum* e integra-los nas equipas para que não só ajudem as equipas a adotar o novo processo mas aprender com essa adoção, podendo partilha-la com as restantes equipas. Os *coaches* deverão focar a sua passagem de conhecimento nos princípios e valores *Agile*, potenciando assim este fator de sucesso da adoção do novo processo, realizando em particular nas seguintes atividades:
    - Participar nas reuniões de Planeamento do *Sprint*;
    - Participar nas reuniões de Revisão do *Sprint*;
    - Participar nas reuniões da Retrospetiva do *Sprint*;
    - Realizar *mentoring* ao *Scrum Master* e ao *Product Owner;*
    - Centrar a sua participação na observação, *feedback* e resposta a questões.

    Para além das atividades referidas anteriormente, o *Coach* deverá ajudar as equipas de forma individual a implementar as seguintes recomendações da Tabela 17:
    - Encorajar a autoatribuição de tarefas;
    - Reintroduzir papéis específicos;
    - Alinhar os incentivos existentes com os futuros;
    - Utilização de documentação apropriada que facilite a comunicação;
    - Focar a responsabilidade na equipa e não no individuo;
    - Realizar uma boa e moderada utilização dos especialistas;
    - Paralelizar tarefas da equipa;
    - Aumentar e partilhar conhecimento constantemente;
    - Participação ativa e constante do cliente no processo de desenvolvimento;
    - Introduzir alterações nas equipas, alterando contentores, atenuando diferenças ou alterando formas de interação entre os elementos da equipa;
    - Construir um ambiente de partilha e aprendizagem para dar à equipa capacidade de tomada de decisão;
    - Implementar um sistema democrático de voto;
    - O gestor de projeto deverá desempenhar um papel de facilitador.

5. **Realização de projetos piloto.** Encontra-se intimamente associado ao tipo de transição *Start Small*, utilizado pela empresa Yahoo! no ano de 2005 (Benefield, 2008). Tem como linhas de ação a criação de um conjunto representativo de projetos piloto, aprender com os

mesmos, e então disseminar o *Agile* faseadamente pela organização. Trata-se da forma de transição mais comum na transição para processos de desenvolvimento *Agile*. Tem custos mais baixos, sucesso mais cedo, menos risco, menos *stress* organizacional e não carece de uma reorganização inicial da organização, a qual poderá ser feita mais à frente no tempo. Os projetos piloto permitem obter a informação necessária para a adaptação do novo processo à realidade, antes de o divulgarmos por toda a organização. Quando têm sucesso, permitem motivar as pessoas, servindo de exemplo ao que lhes vai acontecer (Conboy *et al.*, 2010). Permitem que as pessoas tenham uma visão do seu futuro, com exemplos práticos e reais. É importante que os pilotos realizados tenham características e necessidades de desenvolvimento diferentes, para que seja coberto um maior número de cenários possível.

A definição da versão base do futuro processo de desenvolvimento de *software* da CGD e o conjunto de medidas a aplicar influenciadores dos fatores de sucesso de adoção do *Scrum*, respondem às questões deste trabalho de investigação. No próximo capítulo será abordada a implementação desta solução.

# 8 IMPLEMENTAÇÃO

Este capítulo aborda os principais aspetos referentes à implementação da solução apresentada na secção anterior. São endereçadas cinco vertentes: (i) Organização e responsabilidades no desenvolvimento do projeto; (ii) Cronograma de implementação do projeto; (iii) Custos de Implementação; (iv) Avaliação da implementação; (v) Valor acrescentado para a organização;

## 8.1 Organização do projeto

A organização do projeto adotada considera dois níveis de responsabilidade: (i) Liderança; (ii) Operacional. Encontrando-se o nível de Liderança mais focado na estratégia e o nível Operacional na execução da solução.

Ao nível da **Liderança** deverá ser constituído um Comité de Acompanhamento, presidido pelo Administrador do SSI. Este órgão tem como participantes os responsáveis das Direções envolvidas e a gestão do projeto, envolvendo desta forma os principais responsáveis de cada Direção. Reunindo mensalmente, o Comité de Acompanhamento, tem como principais competências as seguintes:

- Definição da estratégia do projeto;
- Acompanhamento global da evolução do projeto;
- Aprovação das alterações ao plano;
- Resolução de problemas escalados pelo nível Operacional/Execução.

Ao nível **Operacional**, focada na execução do projeto, é necessária a existência de duas equipas: (a) Grupo de Trabalho, (b) Equipa de Gestão. O Grupo de Trabalho é constituído pelos responsáveis de segunda linha das direções, os Coordenadores das Unidades, pelos *coaches* e a equipa de gestão do projeto. Representa desta forma os responsáveis de segunda linha de cada Direção envolvida. Esta equipa tem as seguintes responsabilidades:

- Garantir a uniformização e implementação do novo processo de desenvolvimento;
- Aprovar alterações ao processo com base na prática;
- Acompanhar os trabalhos semanais;
- Dar parecer sobre alterações ao plano.

A Equipa de Gestão do projeto, constituída pelo gestor de projeto e pelo responsável operacional, tem as seguintes responsabilidades:

- Gerir, executar e monitorizar o progresso do plano do projeto;
- Assegurar a articulação entre as várias pessoas e equipas envolvidas;

- Garantir a aprovação e execução dos planos de operacionalização;
- Resolver ou escalar problemas que coloquem em causa o objetivo do projeto;

## 8.2 Cronograma

Para se poder dar seguimento à implementação existe a necessidade de um cronograma, apresentado na Figura 8 abaixo, no qual são identificadas as principais atividades da adoção do novo processo de desenvolvimento baseado em *Scrum*, na CGD.

O cronograma e as ações apresentadas no mesmo têm em conta a solução a adotar, nomeadamente as medidas que influenciam o sucesso da adoção do *Scrum*:

(i) Plano de comunicação eficaz;

(ii) Envolvimento das pessoas;

(iii) Plano de formação para os internos e externos que aborde as componentes de *Scrum*, Técnicas de Desenvolvimento, Sociais e de negócio;

(iv) *Coaching*;

(v) Realização de projetos piloto.

No cronograma, considerando o plano de comunicação, são realizadas as seguintes sessões:

- ***Kickoff* do projeto.** Presidido pela Administração, tem como objetivo dar início ao projeto e comunicar as razões da mudança a todos os colaboradores;
- **Sessões de comunicação às equipas**. Permitem que os superiores hierárquicos diretos possam comunicar como esta mudança deverá afetar cada um dos colaboradores das suas equipas;
- **Sessões de partilha de experiências**. São uma oportunidade de partilha de experiências entre pares;
- **Sessão de encerramento**. Presidida pela Administração, permite apresentar os resultados do projeto, lições aprendidas e encerrar o projeto.

O *Kickoff* do projeto e a Sessão de comunicação às equipas é realizada no início do projeto, sendo a partilha das experiências apenas realizada no decorrer dos primeiros pilotos. A Sessão de encerramento será a última atividade a realizar no projeto.

No que diz respeito ao envolvimento das pessoas, são realizadas as seguintes atividades:

- **Comité de Acompanhamento mensal**. Envolve os responsáveis de todas as Direções, com uma periodicidade mensal;

- **Comité Operacional**. Envolve os responsáveis de 2ª linha das Direções, com uma periodicidade semanal.
- **Desenho inicial do processo de desenvolvimento de *software***. Serão realizadas um conjunto de sessões de trabalho para desenho do futuro processo de desenvolvimento de *software* partindo da versão base da solução a adotar. Envolve representantes de várias equipas de desenvolvimento.
- **Adaptações ao processo de desenvolvimento de *software***. Esta atividade é a continuação da atividade anterior, após a realização dos primeiros pilotos, trazendo com a experiência obtida, novos contributos para um melhor desenho de processo final.

No âmbito da formação, o cronograma, contém as seguintes atividades:

- **Definição do plano global de formação**. Serão realizadas um conjunto de sessões com a Direção responsável pela formação e um conjunto de representantes das direções envolvidas para ser definido um plano global de formação para os colaboradores internos e externos, que aborde as componentes de *Scrum*, Técnicas de Desenvolvimento, competências Sociais e de Negócio;
- **Formação dos *Coaches.*** Esta formação permitirá aos *coaches* aprofundarem o conhecimento sobre o novo processo de *software* e a compreensão dos valores e os princípios *Agile*;
- **Formação dos elementos das equipas do piloto.** Será ministrada formação *Agile Scrum* a todos os elementos das equipas piloto.
- **Formação das equipas de *Rollout.*** Será ministrada formação *Agile Scrum* a todos os elementos das equipas de *Rollout*.

O *Coaching* trata-se de uma atividade constante no cronograma, a partir da entrada da primeira equipa em piloto.

Por último temos as atividades de piloto de *rollout.* O objetivo é inicialmente entrar em piloto com dez equipas de desenvolvimento, preferencialmente que trabalhem com tecnologias e clientes diferentes. Terminado o piloto das 10 equipas e depois das adaptações ao futuro processo de desenvolvimento de *software*, provenientes da experiência obtida, inicia-se o *rollout*, composto por uma vaga de vinte equipas, e duas vagas de trinta equipas, perfazendo um total de 3 vagas.

Prevê-se a conclusão do projeto em 12 meses.

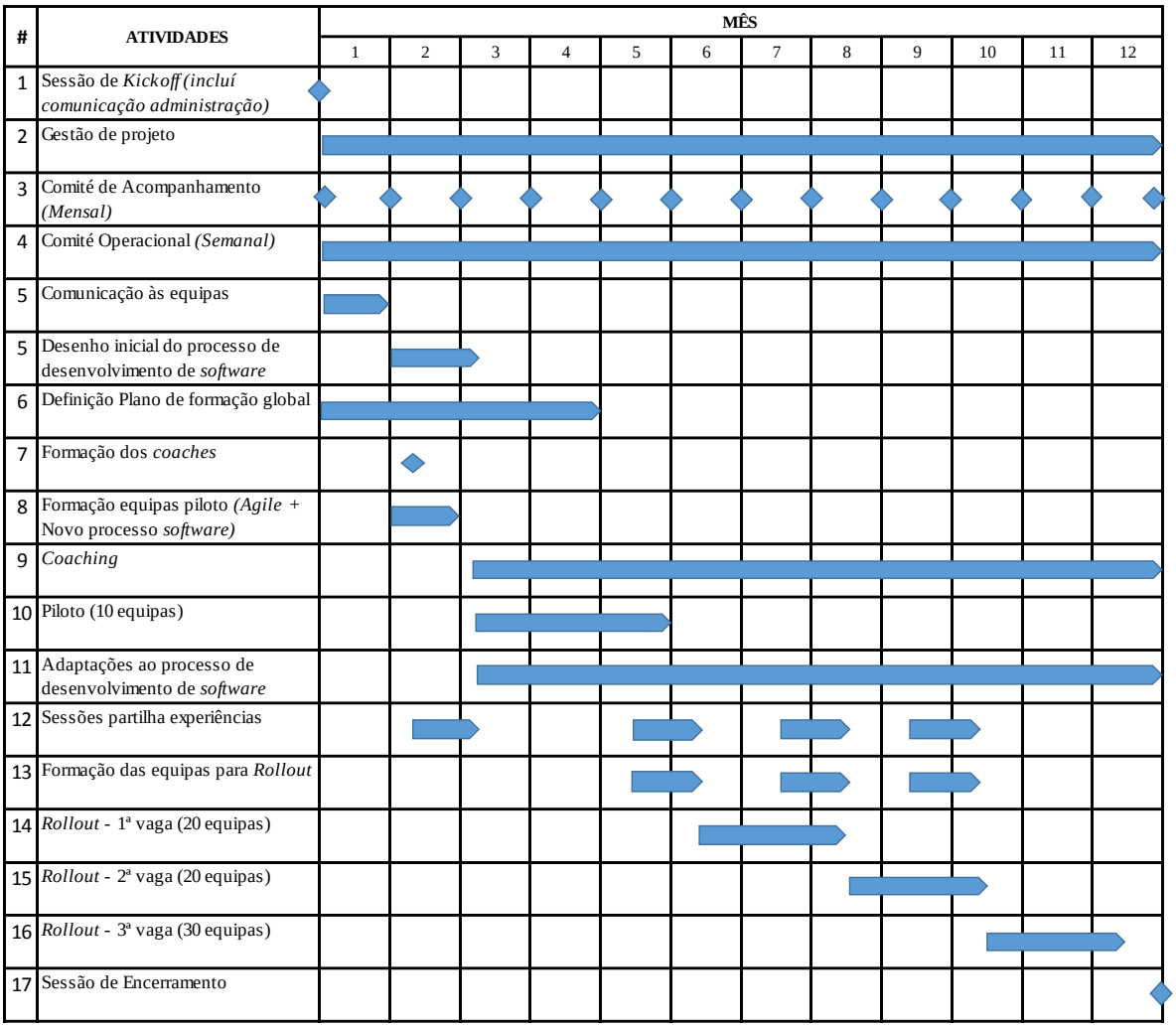


**Figura 8 - Cronograma do projeto**

Fonte: Elaboração própria.

## 8.3 Custos de Implementação

No que diz respeito aos custos de implementação, a Tabela 18, apresenta os custos de cada componente do projeto, com base nos seguintes pressupostos:

1. O projeto de implementação é totalmente realizado por recursos internos. Não existe participação de entidades externas à CGD.
2. Equipa de gestão de projeto é constituída por um gestor de projeto, um responsável operacional e um recurso operacional. O custo interno de um gestor de projeto sénior na CGD e de um responsável operacional sénior é de 60€/hora. O recurso operacional interno tem um custo de 50€/hora. A alocação do gestor de projeto é de 40% e a do responsável e recurso operacional é de 100%.

3. Serão necessários 6 *coaches* para acompanhar em simultâneo 5 equipas cada. Este é o número necessário para que durante o piloto e em cada vaga de *rollout* existam *coaches* suficientes para acompanhar semanalmente as equipas, 8 horas por semana.
4. Serão realizadas 12 sessões de 2 horas de partilha de experiências. Estas sessões no total deverão contar com a participação dos 640 colaboradores internos e externos à CGD, os quais participam ativamente nas equipas de desenvolvimento de *software*. O custo médio dos participantes nestas sessões é de 45€/hora.
5. O desenho do processo deverá ser consolidado em 12 sessões de trabalho de 2 horas cada, ao longo do tempo de projeto. Conta com a participação de 12 pessoas, com algum nível de senioridade, com um custo médio de 60€/hora.
6. A definição dos planos de formação na CGD encontra-se a cargo da Direção de Pessoal. As sessões realizadas no âmbito deste projeto servem para esclarecer e definir em traços gerais o plano de formação pretendido. Serão realizadas 2 sessões de 2 horas, com as 10 pessoas presentes no comité operacional. O seu custo hora médio é de 60€.
7. O comité de acompanhamento é composto por 12 sessões de 10 pessoas ao nível de direção. O seu custo hora médio é de 70€.
8. O comité operacional reúne-se com uma periodicidade semanal, com sessões de 1 hora, ao longo dos 12 meses de projeto. O custo médio dos participantes é de 60€/hora. As sessões têm a participação de 10 pessoas aproximadamente.
9. Para a formação de *Agile/Scrum* são necessários 2 formadores, dado que a formação será faseada ao longo do tempo. Serão realizadas 18 sessões de formação de 14 horas cada, semelhantes às sessões que hoje existem no plano de formação da CGD.
10. Formação *Agile/Scrum* conta com 540 colaboradores (100 colaboradores já se encontram formados) a um rate médio de 50€/hora.

O custo global do projeto é de **1.171.340 €**, sendo que 645.260 € são gastos com a equipa dedicada ao projeto, 44.400 € relativos à participação em comités, 73.440 € com participações em sessões de trabalho e 408.240 € relativos à participação em formação.

**Tabela 18 - Custos de Implementação do novo processo de *software***

Fonte: Elaboração própria

| Componente | Custo (s/IVA) | Observações |
|---|---|---|
| **Equipa de Gestão** | **183.260 €** | |

| Componente | Custo (s/IVA) | Observações |
|---|---|---|
| *Gestão projeto* | *36.960 €* | *Alocação de 40%* |
| *Responsável operacional* | *77.000 €* | *Alocação de 100%* |
| *Recurso Operacional* | *69.300 €* | *Alocação de 100%* |
| ***Coaches*** | **462.000 €** | 6 *coaches* |
| **Subtotal Equipa Operacional** | **645.260 €** | |
| **Comité Acompanhamento** | **8.400 €** | 12 sessões (1h) |
| **Comité Operacional** | **36.000 €** | 50 sessões (1h) |
| **Subtotal Comités** | **44.400 €** | |
| **Sessões partilha experiências** | **57.600 €** | 12 sessões (2h) |
| **Sessões desenho do processo** | **14.400 €** | 12 sessões (2h) |
| **Sessões definição plano de formação** | **1.440 €** | 2 sessões (1h) |
| **Subtotal Sessões trabalho** | **73.440 €** | |
| **Equipa formadores *Agile/Scrum*** | **30.240 €** | 2 formadores/18 sessões |
| **Participação Formação *Agile/Scrum*** | **378.000 €** | 18 sessões; 252 h formação |
| **Subtotal Formação** | **408.240 €** | |
| **TOTAL** | **1.171.340 €** | |

## 8.4 Avaliação da implementação

Ao longo do projeto de implementação, o mesmo deverá ser monitorizado, com o objetivo de avaliar a sua execução. Mensalmente para além do orçamento do projeto deverá ser acompanhado o cumprimento dos seguintes *milestones* do projeto apresentados na Tabela 19, de acordo com o cronograma definido anteriormente na Figura 8.

**Tabela 19 - *Milestones* do projeto de implementação**

Fonte: Elaboração própria

| Objetivo | *Milestone* | Data |
|---|---|---|
| **Piloto (10 equipas)** | Comunicação às equipas | Final 1º mês |
| | Versão inicial do processo de *software* | Final 1ª semana, 3º mês |
| | Formação dos *Coaches* | Início 2ª semana, 2º mês |
| | Formação das equipas piloto | Final 1ª semana, 3º mês |
| | 1ª e 2ª Sessão de partilha de experiências | Final 1ª semana, 3º mês |

| Objetivo | *Milestone* | Data |
|---|---|---|
| | Início do Piloto com 10 equipas | Final 1ª semana, 3º mês |
| | 2ª Versão do processo de *software* | Meio 5º mês |
| **1ª Vaga *Rollout*** | Formação equipas da 1ª vaga de *rollout* | Meio 6º mês |
| | 3ª, 4ª e 5ª Sessão de partilha de experiências | Meio 6º mês |
| | Início 1ª vaga *rollout* | Meio 6º mês |
| | 3ª Versão do processo de *software* | Início 8º mês |
| **2ª Vaga *Rollout*** | Formação equipas da 2ª vaga de *rollout* | Meio 8º mês |
| | 6ª, 7ª e 8ª Sessão de partilha de experiências | Meio 8º mês |
| | Início 2ª vaga *rollout* | Meio 8º mês |
| | 4ª Versão do processo de *software* | Início 10º mês |
| **3ª Vaga *Rollout*** | Formação equipas da 3ª vaga de *rollout* | Meio 10º mês |
| | 9ª, 10ª, 11ª e 12ª Sessão de partilha de experiências | Meio 10º mês |
| | Início 3ª vaga *rollout* | Meio 10º mês |
| | Versão final do processo de *software* | Final 2ª semana, 12º mês |
| **Encerramento** | Sessão de encerramento do projeto | Final 12º mês |

Realizada a avaliação orçamental e do cumprimento dos *milestones* do projeto de implementação, importa avaliar os fatores de sucesso afetados durante esta fase, nomeadamente: (i) Motivação das pessoas; (ii) Compreensão dos valores e princípios *Agile*; (iii) Trabalho em equipa coerente e auto-organizado; (iv) Envolvimento do Cliente; (v) Tomada de decisão descentralizada. Estes fatores deverão ser avaliados de forma qualitativa a partir da qual se consiga aferir o estado de cada um dos critérios a avaliar. Esta avaliação poderá ser realizada com base em inquéritos e aferida também pelos *coaches* junto das equipas de desenvolvimento.

## 8.5 Valor acrescentado

Com a adoção do modelo *Scrum* ao processo de desenvolvimento de *software* da CGD, importa identificar o valor que esta implementação poderá trazer para a organização. O valor acrescentado do novo processo de desenvolvimento de *software* da CGD assenta nos seguintes pontos:

- **Maior produtividade e menores custos**. As equipas que utilizam processos *Agile* são 16% mais produtivas do que a média da indústria (Mah, 2008). A auto-organização das equipas é um dos fatores que influência este aumento de produtividade. Assim que as equipas se tornam mais produtivas, os custos acabam por diminuir.
- **Entregas de valor para o negócio mais rápidas**. Os projetos baseados em processos *Agile* têm um *time to market* 37% acima da média da indústria (Mah, 2008). Um dos objetivos dos modelos *Agile* é a entrega de valor para o Cliente. O artefacto *Backlog* de produto encontra-se organizado para que as primeiras atividades realizadas e entregues sejam as que representem maior valor para o cliente.
- **Melhoria no compromisso e satisfação dos colaboradores.** Nas organizações, a percentagem de colaboradores satisfeitos aumenta para o dobro (Fry and Greene, 2007). A satisfação aumenta com o objetivo comum e com o conhecimento do todo.
- **Maior qualidade.** Em 51 estudos a diferentes projetos, David Rico (2008), encontrou um aumento mínimo de 10% na qualidade. As entregas parciais assentes sobre o conhecimento da equipa levam a um aumento de qualidade.

# 9 CONCLUSÕES

Este trabalho de investigação teve como principal preocupação responder aos desafios da aplicação do modelo *Scrum* ao processo de desenvolvimento de *software* da Caixa Geral de Depósitos (CGD). A resposta focou-se na definição de um novo processo de desenvolvimento de *software*, com base no modelo *Scrum* e num conjunto de medidas que influenciam positivamente os fatores de sucesso na adoção novo modelo. Neste âmbito foram formuladas as respostas às questões da investigação:

1. Definição e caracterização do futuro processo de desenvolvimento de *software* baseado no modelo *Scrum*;
2. Medidas a realizar para que a adoção do *Scrum* ocorra com sucesso.

Iniciou-se um diagnóstico situacional, o qual permite identificar o atual processo de desenvolvimento de *software*, a experiência da CGD com a utilização do *Scrum* e o estado em que se encontram os fatores identificados como influenciadores do sucesso da aplicação do modelo *Scrum* ao processo de desenvolvimento de *software*. Com base neste diagnóstico, a solução a implementar foca-se em 2 vertentes: (i) Futuro Processo de Desenvolvimento de *Software*; (ii) Medidas a aplicar, influenciadoras do sucesso da aplicação do *Scrum*. Adicionalmente é caracterizado o projeto para a implementação da solução apresentada.

Nesta secção procede-se às conclusões finais sobre a resposta produzida às questões de investigação formuladas.

## 9.1 Limitações

Neste trabalho de investigação destacam-se as seguintes limitações:

- **Os custos apresentados no trabalho de investigação são indicativos.** Por razões de confidencialidade não são divulgados os custos hora reais dos colaboradores da CGD.
- **Falta de informação quantitativa sobre trabalho das equipas *Scrum*.** Não existe informação quantitativa sobre o trabalho que tem vindo a ser realizado pelas equipas de *Scrum* que se possa utilizar para que sejam demonstrados os benefícios da utilização do *Scrum* nos projetos existentes.
- **Não foi aprofundado o tema do modelo de contratação de recursos humanos externos no que respeita ao plano de formação dos mesmos**. Representando a maioria dos colaboradores das áreas de desenvolvimentos de *software*, a CGD necessita de rever a influência que tem relativamente ao alinhamento do plano de formação de cada colaborador externo.

- **A solução apresentada é específica para a CGD**. Para efeitos de generalização da solução apresentada, é requerida uma adaptação à realidade da organização à qual se pretende aplicar.

## 9.2 Futuro processo de desenvolvimento de *software*

Relativamente ao futuro processo de desenvolvimento de *software* destacam-se as seguintes características:

- **Baseado no modelo *Scrum*.** Objetivo inicial deste trabalho de investigação, basear o futuro processo de desenvolvimento de *software* no modelo *Scrum*, cumprindo os valores e princípios *Agile*, nomeadamente: (i) Indivíduos e interações em vez de processos e ferramentas; (ii) *Software* a funcionar em vez de documentação; (iii) Colaboração do cliente em vez de negociação de contratos; (iv) Resposta à mudança em vez de seguir um plano.
- ***Waterfall-up-front*.** As regras de governação da CGD, pela dimensão da organização, requerem que sejam definidos os requisitos e os planos antes de se iniciar o trabalho de desenvolvimento. Estes planos formam o contrato entre o negócio e os sistemas de informação, o qual define a direção do projeto, a sua *timeline* e o seu orçamento.
- **Desenhado pelas pessoas para a organização**. Não só tendo implicações na aceitação do mesmo por parte das pessoas, como levando-o a adaptar-se à realidade e aos problemas encontrados no dia-a-dia.
- **Baseado na experiência com *Scrum*.** A CGD cria um processo de desenvolvimento de *software* baseado na experiência diária das equipas que desde do ano de 2010 utilizam o modelo *Scrum* no seu processo de desenvolvimento.
- **Adapta os papéis e responsabilidades antigos.** Neste tipo de transições tem que se assegurar que as pessoas que desempenham determinado papel ou tenham determinada responsabilidade no atual processo de desenvolvimento, saibam o papel ou a responsabilidade que vão ter no futuro;
- Apresenta um conjunto de vantagens, tais como:
    - **Aumento da produtividade e diminuição dos custos**. As equipas que adotam modelos *Agile* são no mínimo 16% mais produtivas do que a média da indústria que tem por base os modelos Tradicionais;
    - **Entregas de valor para o negócio mais rápidas.** As equipas que adotam modelos *Agile* têm um *time-to-market* 37% acima da média da indústria que tem por base os modelos Tradicionais;

- **Melhoria no compromisso e satisfação dos colaboradores.** As equipas que adotam modelos *Agile* aumentam em média para o dobro a satisfação dos seus colaboradores;
- **Maior qualidade da entrega.** As equipas que adotam modelos *Agile* têm um aumento da qualidade do produto de *software* entregue 10% acima da média da indústria que tem por base os modelos Tradicionais;

## 9.3 Medidas influenciadoras do sucesso da adoção do *Scrum*

No que respeita às medidas influenciadores do sucesso da adoção do *Scrum* destacam-se as seguintes medidas/conclusões:

1. **Necessária a existência de um Plano de comunicação eficaz.** A comunicação sobre que se pretende realizar, assim como a informação sobre o estado da mudança, eleva a motivação e a aceitação do novo modelo de desenvolvimento por parte dos colaboradores da organização. Conforme as entrevistas realizadas, comparando com outras alterações processuais da CGD, é considerada uma boa prática a partilha e a comunicação para que exista um envolvimento das pessoas. A comunicação deve ter em conta as razões da mudança e qual o impacto individual na pessoas. A criação de oportunidades para os pares partilharem as suas experiências é um elemento fundamental para que a mensagem passe, principalmente a partilha de experiências positivas. Na fase de implementação este plano de comunicação é colocado em prática através das seguintes sessões: (i) ***Kickoff* do projeto.** Presidido pela Administração, tem como objetivo dar início ao projeto e comunicar as razões da mudança a todos os colaboradores; (ii) **Comunicação às equipas**. Permitem que os superiores hierárquicos diretos possam comunicar como esta mudança deverá afetar cada um dos colaboradores das suas equipas; (iii) **Partilha de experiências**. São uma oportunidade de partilha de experiências entre pares dentro da CGD; (iv) **Sessão de encerramento**. Presidida pela Administração, permite apresentar os resultados do projeto, lições aprendidas e encerrar o projeto.
2. **Envolvimento das pessoas.** O envolvimento das pessoas, principalmente dos mais céticos, melhora os níveis de aceitação do novo processo de desenvolvimento de *software*. O desenho deverá ser realizado pelas pessoas e corrigido ao longo do tempo, com base nas experiências vividas em cada projeto. Representantes dos principais intervenientes do processo de desenvolvimento, incluindo clientes, deverão ser envolvidos no desenho do novo processo e participar na implementação. O

envolvimento transversal da organização é muito importante para a existência *buy-in* desta mudança. Para além deste *buy-in*, a existência de múltiplas realidades no atual processo de desenvolvimento, que devem ser escutadas e incluídas ou adaptadas ao novo processo de desenvolvimento de *software*. Para colocar em prática são realizadas as seguintes atividades na implementação do novo processo de desenvolvimento de *software*: (i) **Comité de Acompanhamento mensal**. Envolve os responsáveis de todas as Direções, com uma periodicidade mensal; (ii) **Comité Operacional**. Envolve os responsáveis de 2ª linha das Direções, com uma periodicidade semanal. (iii) **Desenho inicial do processo de desenvolvimento de *software***. Serão realizadas um conjunto de sessões de trabalho para desenho do futuro processo de desenvolvimento de *software* partindo da versão base da solução a adotar. Envolve representantes de várias equipas de desenvolvimento. (iv) **Adaptações ao processo de desenvolvimento de *software***. Esta atividade é a continuação da atividade anterior, após a realização dos primeiros pilotos, trazendo com a experiência obtida, novos contributos para um melhor desenho de processo final.

3. **Plano de formação para os colaboradores internos e externos que aborde as componentes de *Scrum*, Técnicas de Desenvolvimento, competências Sociais e de Negócio.** A formação das pessoas desempenha um papel bastante importante na adoção de processos baseados no modelo *Scrum*, potenciando um vasto conjunto de fatores de sucesso, nomeadamente: a aprendizagem dos princípios e valores *Agile*, o aumento das competências de desenvolvimento, sociais e o conhecimento do negócio. A formação a realizar deverá focar o seguinte: (i) *Agile*/*Scrum*; (ii) Técnicas de desenvolvimento; (iii) Técnicas de comunicação; (iv) Conceitos do negócio. Durante a fase de implementação será dada a formação em *Agile/Scrum* a todos os participantes do processo de desenvolvimento de *software* da CGD. Relativamente às restantes formações serão incluídas num plano de formação a ser definido durante o projeto. Para que as formações incluam os colaboradores internos e externos da CGD recomenda-se a revisão do modelo de contratação de colaboradores externos para que a CGD tenha uma maior influência no plano de formação dos mesmos, enquadrando-os com as suas necessidades.
4. ***Coaching***. O *coaching* é fundamental para a correta adoção do novo processo de desenvolvimento de *software*, em particular na compreensão dos valores e princípios *Agile*. O objetivo é formar um conjunto de *coaches* em *Scrum* e integra-los nas equipas

para que não só ajudem as equipas a adotar o novo processo mas aprender com essa adoção, podendo partilha-la com as restantes equipas. Os *coaches* deverão focar a sua passagem de conhecimento nos princípios e valores *Agile*, potenciando assim este fator de sucesso da adoção do novo processo. Os *coaches* vão dedicar-se a 5 equipas de cada vez, durante a fase de implementação, trabalhando 8 horas por semana com cada uma.

5. **Realização de projetos piloto.** O objetivo é a criação de um conjunto representativo de projetos piloto, aprender com os mesmos, e então disseminar o *Agile* faseadamente pela organização. O sucesso permite motivar as pessoas apresentando uma visão do seu futuro, com exemplos práticos e reais. É importante que os pilotos realizados tenham características e necessidades de desenvolvimento diferentes, para que seja coberto um maior número de cenários possível. Na implementação serão realizados 10 pilotos, com equipas que revelem características e realidades diferentes em termos tecnológicos e tipo de negócio, por forma a abranger o maior número de casos possíveis.

## 9.4 Contributo para a Caixa Geral de Depósitos

Este trabalho de investigação, é um exemplo de como o mundo académico pode apoiar as organizações, contribuindo para a definição do projeto de aplicação do *Scrum* ao processo de desenvolvimento de *software* da Caixa Geral de Depósitos, fornecendo uma base sustentada a partir da qual a organização poderá trabalhar, aplicando um conjunto de medidas que tem como objetivo influenciar positivamente o sucesso do mesmo.

# 10 BIBLIOGRAFIA

Beck., K., & Beedle., M., & Bennekum., A., & Cockburn., A., & Cunningham., W., & Fowler., M., & Grenning., J., & Highsmith., J., & Hunt., A., & Jeffries., R., & Kern., J., & Marick., B., & Martin., R., & Mellor., S., & Schwaber., K., & Sutherland., J., & Thomas., D. 2001. ***Agile manifesto*.** http://*Agile*manifesto.org/

Begel, A., & Nagappan, N. 2007. Usage and perceptions of agile software development in an industrial context: An exploratory study. ***Proceedings of first international symposium on empirical software engineering and metrics***:255-264. Washington, DC: IEEE Computer Society.

Benefield, G. 2008. Rolling out agile in a large enterprise. ***Proceedings of the 41st annual hawaii international conference on system sciences***, 461-470.

Bohem, B. 2002. Get ready for agile methods, with care. ***IEEE Computer Society***, 35(1):64-69.

Boehm, B., & Turner, R. 2004. ***Balancing agility and discipline: A guide for the preplexed***. **Addison-Wesley.** Boston, MA: Addison-Wesley.

Boehm, B., & Turner, R. 2005. Management challenges to implementing agile processes in traditional development organizations. ***IEEE Computer Society***, 22(5): 30-39.

Brugali D., & Scandurra P. 2009. Component-based robotic engineering (Part I). ***Robotics & Automation Magazine, IEEE***, 16(4): 84-96.

Creasey, T. & Hiatt, J. 2007. *Best practices in change management*. Prosci.

Chow, T., & Cao, D. 2008. A survey study of critical success factors in agile software projects. ***The Journal of Systems and Software***, 81(2008): 961–971.

Cockburn, A., & Highsmith, J. 2001. Agile software development: The business of innovation. ***IEEE Computer Society***, 34(11):131-133.

Cockburn, A., & Highsmith, J. 2001. Agile software development: The people factor. ***IEEE Computer Society***, 34(11):131-133.

Cockburn, A. 2008. Using Both Incremental and Iterative Development. ***Crosstalk***, 21(5): 27-30.

Cohn, M. 2010. ***Succeeding with agile: Software development using Scrum*** (1st ed.). Upper Saddle River, NJ: Addison-Wesley.

Cohn, M., & Ford, D. 2003. Introducing an Agile process to an organization [*software* development]. ***Computer***, 36(6): 74-78.

Conboy, K., & Coyle, S., & Wang, X., & Pikkarainen, M. 2010. People over process: Key challenges in agile development. ***IEEE Software***, 28(4):48-57

Deming, W. 1982. ***Out of the Crisis*** (reprinted in paperback by MIT Press, 2003). SPC Press.

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

Freeman, P. 1983. Reusable software engineering: Concepts and research directions. ***ITT Proceddings of the Workshop on Reusability in Programming***, 129-137.

Fry, C., & Greene, S. 2007. Large scale agile transformation in an on-demand world. ***Agile Conference 2007***: 136-142. Washington, DC: IEEE.

Gandomani, T., & Zulzalil, H., & Ghani, A., & Sultan, A., & Parizi, R. 2014. The impact of inadequate and dysfunctional training on agile transformation process: A grounded theory study. ***Information and Software Technology***, 57(2015):295–309.

Greene, B. 2004. Agile methods applied to embedded firmware development. ***Agile Development Conference 2004***: 71-77. Salt Lake City, UT: IEEE.

Highsmith, J. 2003. Agile project management: Principles and tools. ***Cutter Consortium Agile Product & Project Management Executive Report***, 4(2).

Highsmith, J., & Cockburn, A. 2001. Agile software development: the business of innovation. ***IEEE Computer Society***, 34(9):120-127.

Jammalamadaka, K., & Krishna, V. 2013. Agile software development and challenges. ***International Journal of Research in Engineering and Technology (IJRET)***, 2(8):125-129.

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.

Lalsing, V., & Kishnah, S., & Pudaruth, S. 2012. People factors in agile software development and project management. ***International Journal of Software Engineering & Applications (IJSEA)***, 3(1):117-137.

Lawrence, P. 1969. How to deal with resistance to change. ***Harvard Business Review***, January-February, 4-11.

MacCormack, Alan. 2001. Product-development practices that work: How internet companies build software. ***MIT Sloan Management Review***, 42(2):75–84.

Mah, M. 2008. How agile projects measure up, and what this means to you. ***Cutter Consortium Agile Product & Project Management Executive Report***, 9(9).

Mahanti, A. 2006. Challenges in enterprise adoption of agile methods – A survey. ***Journal of Computing and Information Technology***, 14(3):197-206.

Marchenko, A., & Abrahamsson, P. 2008. Scrum in a multiproject environment: An ethnographically-inspired case study on the adoption challenges. ***Agile2008 conference***: 15-26. Toronto, ON: IEEE.

Misra, S. & Kumar, V. & Kumar, U. 2009. Identifying some important success factors in adopting agile software development practices. ***The Journal of Systems and Software***, 82(2009): 1869–1890.

Moore, R., & Reff, K., & Graham, J. & Hackerson, B. 2007. Scrum at a Fortune 500 Manufacturing Company. ***Agile Conference 2007***:175-180.Washington, DC:IEEE.

Nerur, S., & Mahapatra, R., & Mangalaraj, G. 2005. Challenges of migrating to agile methodologies. ***Communications of the ACM***, 48(5):72-78.

Nerur, S. Mahapatra, R. 2001 Revolution or evolution? A Comparison of Object-Oriented and Structured Systems Development

Paasivaara, M., & Lassenius, C. 2014. Communities of practice in a large distributed agile software development organization – Case Ericsson. ***Information and Software Technology***, 56(12):1556-1577.

Paulk, M., & Curtis, B., & Chrissis, M., & Weber, C. 1993. Capability Maturity Model, Version 1.1. ***IEEE Software***, 10(4):18-27.

Petersen, K., & Wohlin, C., & Baca, D. 2009. The *Waterfall* model in large-scale development, ***Product-Focused Software Process Improvement***: 386-400. Berlin: Springer-Verlag.

Royce, W. 1970. Managing the development of large software systems: concepts and techniques. ***IEEE WESCON***, 26(8): 1–9.

Salo, O., & Abrahamsson, P. 2008. Agile methods in european embedded software development organizations: A survey on the actual use and usefulness of extreme programming and *Scrum*. ***IET Software***, 2(1): 58–64.

Schuh, P. 2004. ***Integrating Agile Development in the real world***. Roakland, MA: Charles River Media, Inc..

Schwaber, K., & Beedle, M. 2001. ***Agile software development with Scrum*** (1st ed.). Upper Saddle River, NJ: Prentice Hall.

Schwaber, K., & Sutherland, J. 2013. ***The Scrum guide – The definitive guide to Scrum: the rules of game.*** http://www.*Scrum*guides.org/docs/*Scrum*guide/v1/*Scrum*-Guide-US.pdf

Shewhart, W. 1986. ***Statistical method from the viewpoint of quality control*** (reprint from 1939). Dover.

Sliger, M. 2006. Bridging the gap: Agile projects in the *Waterfall* enterprise. ***Better Software***, July/August, 26-31.

Sommerville, I. 1996. Software process models. ***ACM Computing Surveys***, 28(1):269-271.

Sommerville, I. 2011. ***Software engineering*** (8st ed.). New York: Addison-Wesley.

Takeuchi, H., & Nonaka, I. 1986. The new new product development game. ***Harvard Business Review*** 64(1): 137-146.

Trammell, C., & Pleszkoch, M., & Linger, R., & Hevner, A. 1996. The incremental development process in cleanroom software engineering. ***Decision Support Systems*** (17)1:55–71.

West, D. & Gilpin, M., & Grant, T., & Anderson, A. 2011. ***Water-Scrum-fall is the reality of Agile for most organizations today***. Forrester Research, Inc.

# ANEXOS

## ANEXO I – Matriz de Análise Entrevistas

| Categorias | Questões | Respostas | Citações |
|---|---|---|---|
| Atual processo de desenvolvimento de *software* da CGD | Identifique os principais problemas do atual processo de desenvolvimento? | • Especificação detalhada no início do processo;<br>• Longos ciclos de desenvolvimento;<br>• Muita documentação;<br>• Elevado custo de mudança;<br>• Entrega tardia de valor para o cliente;<br>• Fraco envolvimento do cliente;<br>• Inexistência de visão global na constituição dos requisitos;<br>• Falta de autoridade e capacidade crítica das equipas de desenvolvimento;<br>• Testes somente no final do processo. | " (…) processo exige muita documentação, a qual muitas vezes não é realizada." (E1)<br><br>"O processo de desenvolvimento não está preparado para as constantes alterações que o cliente exige, (…), o cliente só sabe o resultado no final do projeto." (E2)<br><br>"(…) Exigência de especificação completa e detalhada à cabeça, longos ciclos de desenvolvimento, (…) falta de autoridade ou capacidade critica das equipas de desenvolvimento (…) e inexistência de uma visão global na constituição de requisitos." (E4)<br><br>"Temos aplicações que não entram em produção porque estão a aguardar que o cliente realize testes de aceitação." (E5) |
| | Que papéis identifica dentro das equipas de desenvolvimento? | • Gestores de projeto, *Team Leaders*, Programadores, Analistas Funcionais, Responsáveis Técnicos. | "(…) programadores, analistas funcionais (…)" (E2)<br><br>"(…) gestores de projetos, responsáveis técnicos (…)" (E4) |
| | Como se trabalha diariamente com o modelo *Scrum* dentro de uma organização com processo *Waterfall*? | • Utilização do *Scrum* desde 2010;<br>• As equipas cumprem o processo definido pela organização, aplicando o *Scrum* no seu processo interno diário;<br>• Seguem o modelo *Water-Scrum-Fall* descrito por West (2011), adaptando o processo de | "(…) cumprimos com o processo de desenvolvimento do SSI (…), na prática seguimos o modelo *Water-Scrum-Fall* que muitas vezes falámos…(…)"(E1)<br><br>"(…) utilização desde 2010 (…). Muitas vezes sentimos dificuldades em integrar com outras equipas no que respeita a entregas parciais (…)" (E2)<br><br>"(…) somos condicionados pelo ritmo de trabalho externo." (E4) |

| Categorias | Questões | Respostas | Citações |
|---|---|---|---|
| | | desenvolvimento de *software* atual às práticas seguidas pelo modelo *Scrum*;<br>• Dificuldades na integração com outras equipas *Waterfall.* | |
| Motivação/Aceitação alterações ao processo de desenvolvimento | Encontram-se as pessoas motivadas para uma alteração ao processo de desenvolvimento? | • No passado já existiram várias mudanças que não trouxeram benefícios práticos;<br>• As pessoas que frequentaram a formação de *Scrum* mostraram-se bastante motivadas para aplicar um novo modelo ao processo de desenvolvimento da CGD;<br>• A consciência das desvantagens do modelo atual motiva as pessoas à mudança;<br>• Maioria das pessoas acomodadas. | "Os nossos formandos de *Srum* são pessoas altamente motivadas para a mudança" (E1)<br><br>"Perante as desvantagens do processo atual, penso que as pessoas estão abertas a alterações ao processo atual." (E2)<br><br>"Já existiram muitas alterações no SSI que não levaram a nada. Isto torna as pessoas mais passivas. (…) Existe bastante acomodação ao conhecido (…) Os pioneiros são vistos como aves raras." (E4) |
| Aprendizagem dos valores *Agile* | A organização já tem contacto com os modelos *Agile*? | • *Scrum* é utilizado desde 2010 por uma equipa do SSI;<br>• *O Lean* está a ser utilizado por todas as equipas do SSI, nomeadamente com a utilização do Quadros Brancos;<br>• É administrada uma formação interna de *Scrum* desde o ano de 2010. | "A equipa do *Agile* utiliza *Scrum* desde 2010 (…). É dada formação de *Scrum* à CGD, também desde 2010 (…) e já foram formadas mais de 100 pessoas." (E1)<br><br>"O *Lean* tem estado a ser utilizado no SSI nos Quadros Brancos das Unidades." (E5) |
| Competências de Desenvolvimento | Existem organizações com mais especialistas e outras com mais generalistas. Como é realizada a | • Plano de formação para colaboradores internos;<br>• Colaboradores externos são contratados pelas competências que detêm. | "Existem planos de formação para os internos, (…) os externos já são contratados com as competências necessárias (…)" (E1)<br><br>"Há externos a trabalhar na CGD há mais de 10 anos." (E3) |

| Categorias | Questões | Respostas | Citações |
|---|---|---|---|
| | aquisição de competências técnicas e funcionais por parte dos colaboradores nas equipas de desenvolvimento? | • Colaboradores externos tem planos de formação das suas empresas que podem muitas vezes não estar alinhados com a CGD. | "(…) Não existe uma linha de formação para internos e externos. A conjuntura atual ajuda a isto. (…)" (E4) |
| Competências Sociais | Existem várias formas de comunicação formal e informal. De que forma é realizada a comunicação, na equipa, entre equipas e com o cliente? | • Na equipa utiliza-se muito cara a cara;<br>• Para o exterior da equipa, especialmente com o cliente utiliza-se o mail preferencialmente, reuniões com muitas pessoas e documentação extensa;<br>• O *Lean* com os "Quadro Branco" veio criar mais um canal de comunicação entre os elementos das equipas das Unidades. | "(…) utilização excessiva do mail, com conversas extensas (…)" (E1)<br><br>"Os programadores falam muito entre si (…)" (E2)<br><br>"Realizam-se reuniões e reuniões com imensas pessoas (…)" (E4)<br><br>"Os Quadros Branco trouxeram mais uma forma de comunicação nas Unidades." (E5) |
| Conhecimento do negócio | Como é adquirido o conhecimento do negócio por parte das equipas? | • *E-learning* para colaboradores internos;<br>• Experiência adquirida através de contacto com o negócio no dia-a-dia. | "(…) através da formação anual, *e-leaning* (…)" (E1)<br><br>"(…) adquire-se de projeto para projeto (…) os colaboradores externos não são abrangidos pelo *e-learning* (…)"(E2)<br><br>"(…) Não é adquirido de forma sistemática (…)" (E5) |
| Trabalho em equipa | Tipicamente, como se encontram organizadas as equipas de desenvolvimento | • As equipas têm um gestor de projeto ou responsável de projeto, programadores e por vezes um analista funcional; | " (…) constituída por um gestor de projeto, um funcional e programadores (…)" (E3)<br><br>"(…) as tarefas são atribuídas pelo gestor de projeto (…)" (E4) |

| Categorias | Questões | Respostas | Citações |
|---|---|---|---|
| | e como é organizado o seu trabalho? | • Gestor de projeto ou responsável da unidade atribui as tarefas. | |
| Envolvimento do Cliente | Como e quando é envolvido o cliente nos projetos? | • Cliente é envolvido no início e no fim do processo de desenvolvimento;<br>• São realizados pontos de situação para que o cliente saiba o estado do projeto. | "O cliente é envolvido na fase de análise funcional (…)" (E4)<br><br>"(…) realizamos reuniões de *steering* para acompanhamento (…)" (E5) |
| Tomada de decisão | Uma decisão no desenvolvimento pode ser tomada pelo responsável, pela equipa sozinha ou em conjunto com o responsável. Como é feita a tomada de decisão na equipa? As equipas são autónomas? | • Responsável pela equipa toma decisão;<br>• Equipa envolvida na decisão mas nunca na decisão final. | "O responsável da equipa decide."(E1)<br><br>"A equipa decide alguns temas mas o responsável tem a palavra final (…)" (E2)<br><br>"(…) pela cultura da casa as equipas geralmente são externas, logo a decisão não passa por eles (…)" (E3) |

# ANEXO II – Matriz de Entregáveis por fase do processo (1/2)

SISTEMAS DE INFORMAÇÃO sogrupo Grupo Caixa Geral de Depósitos

**Matriz Entregáveis - Fase**

| Fase | Código | Entregável | Fase | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Análise Pedido | Início Projeto | Acomp. | Fecho | Análise funcional | Desenho Técnico | Des. | TI | TA | Formação | Entrada em Produção |
| Análise Pedido | T01.01 | **Caderno de Requisitos do Cliente** | F | | | | | | | | | | |
| | T01.02 | **Informação Complementar de Requisitos** | F | | | | | | | | | | |
| | T01.03 | **Proposta Solução** | F | | | | | | | | | | |
| | | **Orçamento** | F | | | | | | | | | | |
| | T01.05 | **Justificação de Pedidos de Desenvolvimento** | I | | | | | F | | | | | |
| | T01.06 | **Matriz de Tipificação de Pedido** | I | F | | | | | | | | | |
| | T01.07 | **Requisitos para Processos de Circuito OT (orçamentação tab. gerais)** | F | | | | | | | | | | |
| Início do Projeto | T02.01 | **Informação Projeto** | | F | | | | | | | | | |
| | T02.02 | **Documento Definição Projeto (DDP)** | | F | | | | | | | | | |
| | | **Baseline do Projeto** | | F | | | | | | | | | |
| | T02.03 | **Plano Qualidade** | | I | | | | | | | F | | |
| | T03.01 | **Matriz de Riscos e Problemas** | | I | P | F | | | | | | | |
| | F04.01 | **Product BackLog** | | F | | | | | | | | | |
| Acompanhamento | T03.02<br>T03.03 | **Ponto Situação** | | | P | | | | | | | | |
| | T03.08 | **Matriz Entregáveis Projeto** | | I | | F | | | | | | | |
| | F04.02 | **Sprint BackLog** | | | P | | | | F | | | | |
| | F04.03 | **Sprint Burndown Chart** | | | P | | | | F | | | | |
| | F04.04 | **Release BurnDouwn Chart** | | | P | | | | F | | | | |
| Fecho | T10.01 | **Lições Aprendidas** | | I | P | F | | | | | | | |
| | T10.02 | **Inquérito Satisfação Cliente** | | | I | F | | | | | | | |
| | T10.03 | **Inquérito Satisfação Intercalar** | | | | F | | | | | | | |
| | T10.04 | **Relatório Passagem a Manutenção** | | | | F | | | | | | | |

# ANEXO II – Matriz de Entregáveis por fase do processo (2/2)

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Análise Funcional | T04.01 | **AF-Desenho Funcional da Aplicação** | | | | | F | | | | | | |
| | T04.03 | **AF-Desenho Funcional da Transação/Serviço** | | | | | F | | | | | | |
| | T04.06 | **AF-Desenho Funcional do Processo** | | | | | F | | | | | | |
| | T04.08 | **AF-Desenho Funcional de Outputs** | | | | | F | | | | | | |
| | T04.02 | **AF-Inventário Transações/Serviços** | | | | | I | F | | | | | |
| | T04.04 | **AF-Inventário Processos** | | | | | I | F | | | | | |
| | T04.05 | **AF-Inventário de Ficheiros** | | | | | I | F | | | | | |
| | T04.07 | **AF-Inventário de Outputs** | | | | | I | F | | | | | |
| | T04.09 | **AF-Inventário de Tabelas** | | | | | I | F | | | | | |
| | T04.10 | **AF-Definição de Tabelas** | | | | | I | F | | | | | |
| | T04.11 | **AF-Modelo de Dados** | | | | | I | F | | | | | |
| | T04.12 | **AF-Estratégia de Arranque** | | | | | I | | | | F | | |
| | T08.01 | **Estratégia de Formação** | | | | | I | | | | | F | |
| | T07.02 | **Casos Teste_TI_Func_nnn** | | | | | I | | F | | | | |
| | T07.04 | **Lista Testes TI** | | | | | I | | F | | | | |
| | T11.04 | **Lista Testes TA** | | | | | I | | | F | | | |
| Desenho Técnico | T05.01 | **DT-Arquitetura Técnica/Aplicacional do Sistema** | | | | | | F | | | | | |
| | T05.02 | **DT-Desenho Técnico da Funcionalidade** | | | | | | F | | | | | |
| | T05.03 | **DT-Desenho Técnico da Transação/Serviço** | | | | | | F | | | | | |
| | T05.04 | **DT-Macro-fluxo da Aplicação** | | | | | | F | | | | | |
| | T05.05 | **DT-Fluxo da Cadeia** (fluxo automático se aplicável) | | | | | | F | | | | | |
| | T05.06 | **DT-Fluxo de JOB** (fluxo automático se aplicável) | | | | | | F | | | | | |
| | T05.07 | **DT-Definição de Ficheiros** | | | | | | F | | | | | |
| | T05.08 | **DT-Inventário de Tabelas Gerais** | | | | | | F | | | | | |
| | T05.09 | **DT-Requisitos de Gestão de Serviço** | | | | | | F | | | | | |
| | T05.10 | **DT-Inventário de Programas/Módulos** | | | | | | F | | | | | |
| | T03.07 | **Documentos Processo** | | | | | | F | | | | | |
| Desenvolvimento | | **Documento de Passagem de Software** (WEB-REA/RIP) | | | | | | | F | | | | |
| | T06.02 | **Plano de Arranque** | | | | | | | I | | | | F |
| | T07.01 | **Plano Global Testes TI** | | | | | | | F | | | | |
| | | **DT-Especificação Local da Transação DTF** | | | | | | I | F | | | | |
| Testes de Integração | | **Incoerências Detectadas no TI** (ChangePoint) | | | | | | | | F | | | |
| | T11.01 | **Plano Global Testes TA** | | | | | | | | F | | | |
| Testes de Aceitação | | **Incoerências Detectadas no TA** (ChangePoint) | | | | | | | | | F | | |
| Formação | T08.02 | **Manual Utilizador - Índice** | | | | | | | | | | F | |
| | T08.03 | **Manual Utilizador Transação Serviço** | | | | | | | | | | F | |
| | T08.04 | **Manual Utilizador Transação Serviço - Plataforma Balcão** | | | | | | | | | | F | |

**LEGENDA**

| | |
|---|---|
| I | Iniciado |
| F | Fechado |
| P | Periódico |
| | Componente de Gestão |

# ANEXO III – Matriz de Responsabilidades (1/2)

SISTEMAS DE INFORMAÇÃO sogrupo Grupo Caixa Geral de Depósitos

**Matriz de Responsabilidades por Entregável**

| Código | Entregável | Diretor Projeto | Sponsor | Responsável do Pedido | Equipa Projeto / Equipa Subdomínio | Gestor Procura | Equipa Planeamento e Infraestruturas | Equipa TA | Gestão de Níveis Serviço | SSI-Oper Arquitetura Central | Direção de Arquitetura e Estratégia Tecnológica | Qualidade e Melhoria Contínua | Equipa Formação | DCO | Cliente |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **RACI Chart**<br>R = Responsável pela execução<br>A = Aprovar e garantir a execução<br>C = Consultado<br>I = Informado | | | | | | | | | | | | | | |
| T01.01 | **Caderno de Requisitos do Cliente** | | | | C | R | | | I | | | | | | A/R |
| T01.02 | **Informação Complementar de Requisitos** | | | | A/R | I | | | | | C | | | | C |
| T01.03 | **Proposta Solução** | I | I | R* | C | R** | C | C | | | | | | | A |
| | **Orçamento** | I | I | R | C | R** | C | | | | | | | | A |
| T01.05 | **Justificação de Pedidos de Desenvolvimento** | | | | | A/R | | | | | | | | | C |
| T01.06 | **Matriz de Tipificação de Pedido** | I | | A/R | C | C | | | | | I | | | | |
| T01.07 | **Requisitos para Processos de Circuito OT (orçamentação tab. gerais)** | | | | | | | | | R | | | | | |
| T02.01 | **Informação Projeto** | I | A | R | | C | | | | | | | | | I |
| T02.02 | **Documento Definição Projeto (DDP)** | I | I | A/R | | | | | | | | | | | I |
| | **Baseline do Projeto** | I | I | A/R | | | C | C | | | | | C | C | I |
| T02.03 | **Plano Qualidade** | I | | A/R | C | | | | | | | | | | |
| T03.01 | **Matriz de Riscos e Problemas** | I | I | A/R | C | | C | C | | | | | C | | C |
| T03.02<br>T03.03 | **Ponto Situação** | I | I | A/R | C | I | | | | | | | | | I |
| T03.06 | **Pedido de Alteração** | I | I | R | R | I | C | I | | | | | | | A |
| T03.08 | **Matriz Entregáveis Projeto** | | | A/R | | C | | C | C | | | | | C | C |
| T10.01 | **Lições Aprendidas** | I | | A/R | C | C | C | C | C | | | C | C | | |
| T10.02 | **Inquérito Satisfação Cliente** | I | I | I | I | A/R | | | | | | | | | C |
| T10.03 | **Inquérito Satisfação Intercalar** | I | I | I | I | A/R | | | | | | | | | C |
| T10.04 | **Relatório Passagem Manutenção** | I | | R | C | A | | | | | | | | | |
| T04.01 | **AF-Desenho Funcional da Aplicação** | | | A | R | | C | I | | | | | | | I |
| T04.03 | **AF-Desenho Funcional da Transação/Serviço** | | | A | R | | C | I | | | | | | | I |
| T04.06 | **AF-Desenho Funcional do Processo** | | | A | R | | C | I | | | | | | | I |
| T04.08 | **AF-Desenho Funcional de Outputs** | | | A | R | | | I | | | | | | | I |
| T04.02 | **AF-Inventário Transações/Serviços** | | | A | R | | | | | | | | | | |
| T04.04 | **AF-Inventário Processos** | | | A | R | | | | | | | | | | |
| T04.05 | **AF-Inventário de Ficheiros** | | | A | R | | | | | | | | | | |
| T04.07 | **AF-Inventário de Outputs** | | | A | R | | | | | | | | | | |
| T04.09 | **AF-Inventário de Tabelas** | | | A | R | | | | | | | | | | |
| T04.10 | **AF-Definição de Tabelas** | | | A | R | | C | | | | | | | | |
| T04.11 | **AF-Modelo de Dados** | | | A | R | | C | | | | | | | | |
| T04.12 | **AF-Estratégia de Arranque** | I | I | A | R | | C | | | | | | | | I |
| T08.01 | **Estratégia de Formação** | I | I | A | R | | | | | | | | C | | I |
| | **Manual Procedimentos** | | | I | I | | | | | | | | | A/R | I |
| T05.01 | **DT-Arquitetura Técnica/Aplicacional do Sistema** | | | A (a) | R (a) | | C | | | | | | | | |
| T05.02 | **DT-Desenho Técnico da Funcionalidade** | | | A | R | | C | | | | | | | | |
| T05.03 | **DT-Desenho Técnico da Transação/Serviço** | | | A | R | | C | | | | | | | | |
| T05.04 | **DT-Macro-fluxo da Aplicação** | | | A | R | | C | | | | | | | | |
| T05.05 | **DT-Fluxo da Cadeia** (fluxo automático se aplicável) | | | A | R | | C | | | | | | | | |
| T05.06 | **DT-Fluxo de JOB** (fluxo automático se aplicável) | | | A | R | | C | | | | | | | | |
| T05.07 | **DT-Definição de Ficheiros** | | | A | R | | | | | | | | | | |
| T05.08 | **DT-Inventário de Tabelas Gerais** | | | A | R | | | | | | | | | | |
| T05.09 | **DT-Requisitos de Gestão de Serviço** | | | A | R | I | C | | C | | | | | | C |
| T05.10 | **DT-Inventário de Programas/Módulos** | | | A | R | | | | | | | | | | |
| | **Documento de Passagem de Software** (WEB-REA/RIP) | | | A | R | | | | | | | | | | |

# ANEXO III – Matriz de Responsabilidades (2/2)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T07.01 | **Plano Global Testes TI** | | | A | R | | C | | | | | | | | |
| T07.02 | **Casos Teste_TI_Func_nnn** | | | A | R | | C | | | | | | | | |
| | **Incoerências Detectadas no TI** (ChangePoint) | | | A | R | | | | | | | | | | |
| T07.04 | **Lista Testes TI** | | | A | R | | C | | | | | | | | |
| T06.02 | **Plano de Arranque** | I | I | A | R | I | C | I | | | | | | | I |
| T11.01 | **Plano Global Testes TA** | | | C | C | | C | A/R | | | | | | | C |
| | **Incoerências Detectadas no TA** (ChangePoint) | I | I | C | C | | | A/R | | | | | | | C |
| T11.04 | **Lista Testes TA** | | | I | | | | A/R | | | | | | | C |
| | **Requisitos de Níveis de Serviço** | | | C | C | C | | | R | | | | | | A |
| T08.02 | **Manual Utilizador - índice** | | | A | R | | | | | | | | | | I |
| T08.03 | **Manual Utilizador Transação Serviço** | | | A | R | | | | | | | | | | I |
| T08.04 | **Manual Utilizador Transação Serviço - PB** | | | A | R | | | | | | | | | | I |
| | **Manual Formação** | | | C | C | | | | | | | | A/R | | I |

* Se estiver identificado que um determinado Pedido corresponde a um Projeto, poderá ser logo designado um Gestor de Projeto que será responsável pela elaboração da proposta e orçamentação

** A Gestão da Procura é responsável por garantir a execução

c (a) A elaboração do documento poderá ter o apoio do DAT (Direção de Arquitetura Tecnológica)