ISCTE Business School
Instituto Universitário de Lisboa

# MODELO DE GESTÃO DE PROJETOS PARA DIREÇÃO DE IT DE EMPRESA DE SERVIÇOS: DIAGNÓSTICO E PROPOSTA DE MELHORIA

Miguel Prego Gaspar

Projeto de Mestrado

em Gestão

Orientador:

Prof. José Cruz Filipe, Prof. Auxiliar Convidado, ISCTE Business School, Departamento de Marketing, Operações e Gestão Geral

Dezembro 2012

*Esta página foi intencionalmente deixada em branco*

# Índice

Resumo ..... xi

Abstract ..... xiii

Sumário Executivo ..... xv

1 Introdução ..... 1

1.1 Motivação e objetivos ..... 1

1.2 Âmbito da dissertação ..... 2

1.3 Problema a estudar ..... 4

2 Metodologia utilizada ..... 5

2.1 Classificação da metodologia escolhida ..... 5

2.2 Entrevistas semiestruturadas ..... 7

2.3 Revisão de literatura ..... 7

3 Revisão de literatura ..... 9

3.1 As especificidades da gestão de projetos de *software* ..... 9

3.2 Fatores críticos de sucesso para a gestão de projetos de *software* ..... 11

3.3 Modelos de gestão tradicional ..... 16

3.3.1 Apresentação ..... 16

3.3.2 Ciclo de vida ..... 17

3.3.3 Gestão de Recursos Humanos ..... 19

3.3.4 Planeamento do trabalho ..... 19

3.3.5 Execução do trabalho ..... 23

3.3.6 Monitorização e controlo ..... 24

3.3.7 Relatórios e comunicação ..... 25

3.4 Modelos de gestão ágeis ..... 26

3.4.1 Apresentação ..... 27

3.4.2 Ciclo de vida ..... 30

3.4.3 Gestão de Recursos Humanos ..... 32
3.4.4 Planeamento do trabalho ..... 34
3.4.5 Execução ..... 40
3.4.6 Monitorização e controlo ..... 41
3.4.7 Relatórios e comunicação ..... 45
3.4.8 Validação das metodologias ágeis ..... 47
3.5 Os problemas da gestão tradicional do ponto de vista ágil ..... 47
3.6 Conclusões da revisão de literatura ..... 50
4 Apresentação da empresa e do problema em estudo ..... 52
4.1 Descrição da empresa ..... 52
4.2 Descrição da direção de IT ..... 52
4.3 Descrição do ambiente de gestão de projetos ..... 53
4.3.1 Planeamento de Projetos ..... 55
4.3.2 Execução, controlo e monitorização ..... 56
4.3.3 Relatórios e comunicação ..... 56
5 Diagnóstico da situação atual ..... 57
5.1 Resultados das entrevistas ..... 57
5.2 Análise quantitativa ..... 59
5.2.1 Projetos realizados ..... 59
5.2.2 Composição das equipas por projeto ..... 60
5.2.3 Tempo de Análise ..... 61
5.2.4 Tempo de desenvolvimento ..... 62
5.2.5 Tempo de testes ..... 63
5.2.6 Tempos globais ..... 64
5.2.7 Variação do tempo de desenvolvimento face ao estimado ..... 65
6 Propostas de melhorias metodológicas para resolução dos problemas identificados ..... 66
6.1 Recapitulação do problema e objetivos a atingir ..... 66

6.2 Propostas de melhoria ..... 68
6.2.1 Proposta de metodologia ..... 68
6.2.2 Especificidades ao nível da equipa ..... 69
6.2.3 Especificidades ao nível do planeamento ..... 70
6.2.4 Especificidades ao nível da execução ..... 71
6.2.5 Propostas de soluções para os objetivos identificados ..... 72
7 Conclusões ..... 76
8 Possibilidades de trabalho futuro ..... 78
9 Bibliografia ..... 79
10 Anexo A – Questões das entrevistas semiestruturadas ..... 81
11 Anexo B – Dicionário de termos ágeis ..... 82
12 Anexo C – Mapa de respostas às entrevistas de diagnóstico ..... 86
12.1 Dificuldades na fase de Planeamento ..... 86
12.2 Dificuldades nas fases de Execução, Monitorização e Controlo ..... 87
12.3 Dificuldades na Gestão de Recursos Humanos ..... 88
12.4 Necessidades de Informação e Comunicação ..... 89
12.5 Identificação dos pontos fortes da equipa ..... 90
12.6 Identificação dos pontos fracos da equipa ..... 90
12.7 Identificação de sugestões por parte da equipa ..... 91
13 Anexo D – Processos da fase de planeamento definidos pelo PMBOK ..... 92
14 Anexo E – Processos da fase de execução definidos pelo PMBOK ..... 95
15 Anexo F – Processos da fase de Monitorização e controlo definidos pelo PMBOK ..... 96

*Esta página foi intencionalmente deixada em branco*

# Índice de Tabelas

Tabela 1 - Maiores desafios da gestão de projectos de software segundo Thamhain e Wilemon (1986) ..... 10
Tabela 2 - Problemas sentidos pelos gestores e pelos membros de projetos de *software* segundo Hughes e Cotterell (1999) ..... 10
Tabela 3 - Três critérios de sucesso mais referidos para os projetos com sucesso. Estudo de Wateridge (1995) ..... 12
Tabela 4 - Três critérios de sucesso mais referidos para os projectos com falhados. Estudo de Wateridge (1995) ..... 12
Tabela 5 - Três critérios de sucesso mais referidos para todos os projetos. Estudo de Wateridge (1995) ..... 12
Tabela 6 - Processos da fase de planeamento segundo o PMBOK ..... 22
Tabela 7 - Processos da fase de execução segundo o PMBOK ..... 23
Tabela 8 - Processos da fase de monitorização e controlo segundo o PMBOK ..... 24
Tabela 9 - Tipos de relatórios para Miguel (2010) ..... 26
Tabela 10 - Dificuldades identificadas na fase de planeamento ..... 57
Tabela 11 - Dificuldades identificadas na fase de execução, monitorização e controlo ..... 57
Tabela 12 - Dificuldades identificadas ao nível dos Recursos Humanos ..... 58
Tabela 13 - Necessidades de informação e comunicação identificadas ..... 58
Tabela 14 - Pontos fortes da equipa identificados ..... 58
Tabela 15 - Pontos fracos da equipa identificados ..... 58
Tabela 16 - Sugestões de melhoria ..... 58
Tabela 17 - Objectivos de melhoria a atingir ..... 68
Tabela 18 - Proposta de soluções para os objetivos identificados ..... 75

*Esta página foi intencionalmente deixada em branco*

# Índice de Figuras

Figura 1 - Níveis típicos de custos e de esforço ao longo do ciclo de vida dos projetos (adaptado de Miguel 2010) ..... 2
Figura 2 - Esquema da dissertação ..... 3
Figura 3 - Processo de investigação de um caso de estudo baseado em Runeson et al. (2009) . 6
Figura 4 - Critérios de sucesso na gestão de projetos de *software*. Baseado em Wateridge (1995) ..... 13
Figura 5 – Fatores de sucesso segundo Wateridge (1995) ..... 14
Figura 6 - Relações entre as áreas de conhecimento da gestão de projectos. Miguel (2010) . 16
Figura 7 - Estrutura de conhecimentos do PMBOK ..... 17
Figura 8 - Modelo em cascata. Leffingwell (2011) ..... 17
Figura 9 - Responsabilidades do Gestor de Projetos. Baseado no PMBOK ..... 19
Figura 10 - Componentes do Plano de Projeto segundo Wysocky (2003) ..... 20
Figura 11 - Efeitos do planeamento ao longo do tempo para Wysocki (2003) ..... 22
Figura 12 - Princípios fundamentais das metodologias ágeis segundo Leffingwell (2011) .... 27
Figura 13 - Comparação de uso de metodologias ágeis segundo Leffingwell (2011) ..... 28
Figura 14 - O verdadeiro caminho do plano para Rasmusson (2010) ..... 29
Figura 15 - A armadilha do triângulo de ferro no modelo em cascata, segundo Leffingwell (2011) ..... 30
Figura 16 - Ciclo de vida da gestão ágil. Fonte:Rasmusson (2010) ..... 31
Figura 17 - Impacto das metodologias ágeis na equipa. Fonte: Rasmusson (2010) ..... 32
Figura 18 - Horizontes de planeamento ágil. Baseado em Cohn (2005) ..... 35
Figura 19 - Formulação de user stories. Adaptado de Cohn (2005) ..... 35
Figura 20 - Condições de satisfação conduzem o planeamento de *releases* e iterações. Fonte: Cohn (2005) ..... 38
Figura 21 - Processo de estimação de tempos em metodologias ágeis, segundo Cohn (2005) 39
Figura 22 - Os processos do Scrum. (Larman, 2008) ..... 40
Figura 23 - *Release* Burn Down Chart. Adaptado de Cohn (2005) ..... 42
Figura 24 - Previsão de conclusão no Release Burn Down Chart. Fonte: Cohn (2005) ..... 42
Figura 25 - Quadro de tarefas. Fonte: Cohn (2005) ..... 43
Figura 26 -Iteraction Burndown Charts. Fonte: Cohn (2005) ..... 45
Figura 27 - Cronograma ágil. Fonte: Cohn, (2005) ..... 46
Figura 28 - Organigrama da direção de IT da ZTelc ..... 52

Figura 29 - Departamentos operacionais da ZTelc ..... 53
Figura 30 - Departamentos dedicados a projetos na ZTelc ..... 53
Figura 31 - Macroprocesso de gestão de projetos na ZTelc ..... 54
Figura 32 - Processo de desenvolvimento da ZTelc ..... 54
Figura 33 - Estrutura idealizada da equipa de desenvolvimento ..... 54
Figura 34 - Estrutura real da equipa de desenvolvimento ..... 55
Figura 35 - Projectos concluídos / Cancelados ..... 59
Figura 36 - Tempo médio (dias úteis) ..... 59
Figura 37 - Composição das equipas por projeto ..... 60
Figura 38 - Número de projetos por dias úteis de análise ..... 61
Figura 39 - Número de projetos por dias úteis de desenvolvimento ..... 62
Figura 40 - Número de projetos por dias úteis de testes ..... 63
Figura 41 - Número de projetos por dias úteis totais ..... 64
Figura 42 - Duração dos projetos por fase ..... 64
Figura 43 - Relação entre o tempo estimado e o tempo real (dias úteis) ..... 65
Figura 44 - Equipa atual (não ágil) ..... 69
Figura 45 - Equipa proposta (ágil) ..... 70

# Resumo

A gestão de projetos de desenvolvimento de *software* tem apresentado resultados diferentes dos conhecidos para outras áreas, sendo já do senso comum que as suas características são especiais e que por isso a sua gestão oferece desafios suplementares.

Este trabalho começa por explorar que desafios são estes, identificando as suas especificidades e os seus fatores críticos de sucesso através de uma revisão bibliográfica e análise de artigos científicos sobre o tema.

É realizada também uma revisão de literatura acerca das diferentes abordagens existentes para a gestão deste tipo de projetos, fazendo um ponto de situação do estado da arte das duas vertentes mais comuns: a gestão tradicional e a gestão ágil.

Seguidamente é realizado um estudo de caso, analisando a situação real de uma empresa nacional que reconhece a existência de alguns problemas na sua metodologia de gestão de projetos. Esta dissertação realiza um diagnóstico à situação específica desta organização, recorrendo a entrevistas e à análise de alguns dados quantitativos.

Esta dissertação termina com a sugestão de melhorias corretivas para os problemas identificados durante o diagnóstico, fazendo propostas de melhoria para todas as situações identificadas, concluindo com a sugestão de implementação de uma metodologia ágil.

**Palavras-chave:** Gestão de projetos, desenvolvimento de software, metodologias ágeis, Scrum

*Esta página foi intencionalmente deixada em branco*

# Abstract

The projects management of software *development* has been presenting different results from other known areas. It's common sense that its features are special and therefore their management provides additional challenges.

This study begins by exploring what these challenges are, identifying their specificities and their critical success factors through a literature review and analysis of scientific articles about the theme.

Is also made a literature review of the different existing approaches for managing this type of projects, to take stock of the two most common paths: the traditional management and agile management.

Thereafter is performed a case study, analyzing the true situation of a Portuguese company that recognizes the existence of some problems in its project management methodology.

This dissertation conducts a diagnostic of the situation of this organization, based on some interviews and data analysis.

This essay ends with the suggestion of corrective actions to the problems identified, making improvements proposals for all situations identified, ending with the suggestion of implementing an agile methodology.

**Keywords:** Project management, software development, agile methodologies, Scrum

*Esta página foi intencionalmente deixada em branco*

# Sumário Executivo

Este trabalho tem por objetivo diagnosticar o ponto de situação da gestão de projetos na área de desenvolvimento de uma empresa nacional, identificando os problemas existentes e propondo um conjunto de melhorias metodológicas para os corrigir e aumentar o seu desempenho nesta área.

Tendo sido realizado um trabalho preliminar de revisão de literatura para obter toda a informação necessária para este trabalho, o seu conjunto apresenta os seguintes componentes:

**As especificidades da gestão de projetos de *software***

Caracterização das especificidades da gestão de projetos, procurando identificar em que medida a sua gestão é diferente da gestão de outros tipos de projeto e quais são os seus fatores críticos de sucesso.

**Modelos de gestão tradicional**

Explanação do funcionamento das metodologias de gestão de projeto tradicionais, adaptadas para a gestão de projetos de desenvolvimento de *software*. Este módulo apresenta os processos do PMBOK mais relevantes para este estudo, completando-os com informação adicional referente á sua adaptação a este tipo de projetos. A informação é categorizada em apresentação, ciclo de vida, gestão de recursos humanos, planeamento do trabalho, planeamento do trabalho em projetos pequenos, execução do trabalho, monitorização e controlo, relatórios e comunicação.

**Modelos de gestão ágil**

Apresentação das metodologias que se apresentam como alternativas ao modelo de gestão tradicional, defendendo que este modelo não é adequado para a gestão de projetos de desenvolvimento de *software*, e que existe uma alternativa direcionada para as necessidades específicas destes projetos. A informação deste tópico está organizada por: apresentação, vantagens das metodologias ágeis, ciclo de vida, gestão de recursos humanos, planeamento do trabalho, execução, monitorização e controlo, relatórios e comunicação.

**Os problemas da gestão tradicional do ponto de vista ágil**

Visão da metodologia tradicional a partir da metodologia ágil. São apontados os principais problemas reconhecidos e explicada a forma com as metodologias ágeis os abordam.

**Apresentação da empresa em estudo (ZTelc)**

Apresentação da empresa ZTelc, a empresa nacional onde foi realizado o diagnóstico e que é alvo das propostas metodológicas deste trabalho. É feita uma descrição do seu ambiente de gestão de projetos e analisado em detalhe como se processam os projetos a nível de: planeamento, execução, monitorização e controlo, relatórios e comunicação,

**Diagnóstico da situação atual**

São apresentados os resultados do diagnóstico efetuado, apresentando as conclusões obtidas através das entrevistas e relatórios de dados quantitativos.

**Proposta de melhorias metodológicas para resolução dos problemas identificados**

Apresentação resumida de todos os problemas identificados e apresentação de melhorias para os corrigir. São apresentadas algumas especificidades necessárias para que as propostas possam ser colocadas em prática, como a reestruturação da equipa, das fases de planeamento e de execução.

# 1 Introdução

A gestão de projetos de desenvolvimento de *software* envolve um conjunto de complexidades e indeterminações que a tornam mais propensa a erros de planeamento, falhas no controlo durante a execução e inevitavelmente a deslizes dos constrangimentos de âmbito, tempo ou custos.

Esta situação é largamente conhecida, tendo surgido nos últimos anos um conjunto de metodologias novas, especificamente orientadas para a gestão destes projetos, enfrentando as suas necessidades concretas, e tentando minimizar os efeitos menos positivos das metodologias "tradicionais" (Cohn, 2005).

Para além destes problemas, algumas empresas lutam ainda com dificuldades suplementares, como equipas de pequena dimensão com elevada carga de trabalho e a gestão/ execução de atividades de múltiplos projetos em simultâneo (Cohn, 2005).

Esta dissertação propõe-se acompanhar uma equipa de desenvolvimento de uma empresa particular - uma empresa de serviços no sector das telecomunicações - determinando com o apoio da sua direção, os principais problemas e propondo uma abordagem metodológica específica para os resolver. É também proposta uma solução para simplificar/ agilizar os processos de gestão de forma a otimizar ao máximo a sua razão de existir, a criação de código e construção de sistemas e plataformas informáticas, com elevada qualidade, dentro dos constrangimentos planeados.

## 1.1 Motivação e objetivos

O tema desta dissertação foi escolhido essencialmente devido às seguintes motivações:

1. Os projetos de desenvolvimento de *software* são particularmente complexos, dependendo de processos intelectuais de criatividade e imaginação, nem sempre adequados a modelos de gestão tradicional mais formais e burocráticos.
2. Como consequência da identificação do ponto anterior, surgiram nos últimos anos várias metodologias de gestão de projetos, especificamente orientadas para este tipo de desafios, conhecidas por metodologias ágeis, que criam uma nova abordagem. Este trabalho pretende explorar as características das principais metodologias, tradicionais e ágeis e identificar qual é a melhor a solução para um caso específico.
3. Os problemas de gestão de projetos em equipas de desenvolvimento de *software* são partilhados por várias empresas e sectores de atividade. No entanto, a possibilidade de

estudar um caso concreto e propor um conjunto de passos metodológicos para melhorar o seu sucesso na gestão de projetos, focando a atenção num conjunto de questões específico, permite a realização de um trabalho mais focado do que se obteria num estudo genérico e alargado sobre diferentes empresas deste universo.

4. A possibilidade de aliar a dissertação de mestrado às necessidades e objetivos concretos de uma equipa de desenvolvimento de uma empresa nacional real, tendo a hipótese de contribuir para a sua melhoria metodológica na gestão de projetos e em última instância para o sucesso do seu negócio, oferecem um estímulo adicional para a realização deste trabalho.

## 1.2 Âmbito da dissertação

Este trabalho irá focar-se especificamente na gestão de projetos de desenvolvimento de *software*, concentrando-se nas fases operacionais do ciclo de vida do projeto que correspondem às fases de planeamento, execução, monitorização e controlo, que correspondem aos períodos mais longos e com custos mais elevados.

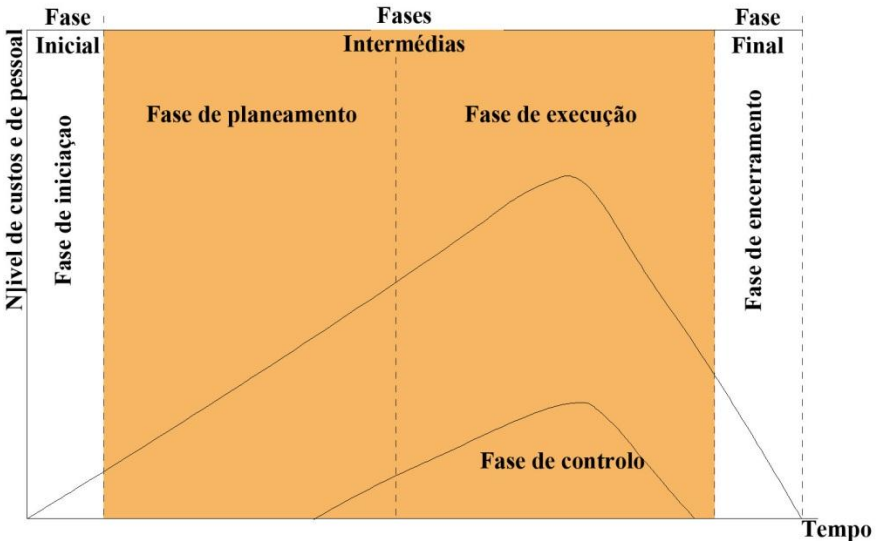



**Figura 1 - Níveis típicos de custos e de esforço ao longo do ciclo de vida dos projetos (adaptado de Miguel 2010)**

Assume-se que:

- As fases que precedem o momento do planeamento foram devidamente realizadas e que todos os projetos estão devidamente avaliados e aprovados para realização.
- As fases que sucedem o período de execução e respetivo controlo não são consideradas relevantes para este trabalho, embora seja entendido que o cumprimento

de processos adequados na fase de encerramento conduza ao fecho adequado de um projeto e que as melhorias significativas nas outras fases de projetos futuros, nomeadamente através da manutenção de dados estatísticos, de lições aprendidas e de afinação das estimativas com base em elementos históricos.

O âmbito desta dissertação será então composto por:

- Revisão de literatura específica para abordagem das questões identificadas nos pontos anteriores e estudo das melhores práticas oferecidas pelas metodologias tradicionais e ágeis para gestão das fases de planeamento, execução e monitorização e controlo.
- Diagnóstico da situação atual no contexto de gestão de projetos na área de IT da empresa em estudo
- Identificação dos problemas específicos na gestão de projetos de desenvolvimento na direção de IT da ZTelc
- Identificação dos pontos a melhorar na direção de IT da ZTelc
- Proposta de melhorias metodológicas de gestão de projetos com base em todos os elementos estudados e trabalhados

De forma resumida, este trabalho inclui os seguintes elementos:

**Figura 2 - Esquema da dissertação**

Embora a empresa seja real, tal como todos os dados trabalhados e estudados, foi solicitada a sua confidencialidade, pelo que neste trabalho será designada por ZTelc.

Devido ao facto de a linguagem ser bastante técnica, e alguns termos não terem tradução para português, serão usadas no decorrer deste trabalho algumas denominações em inglês.

## 1.3 Problema a estudar

Miguel (2010:2) refere "Um gestor já não pode mais criar um cronograma de um projeto apenas através do preenchimento de um ou dois formulários. As empresas exigem muito mais dos seus projetos, do que antes. Uma forma de alcançar um resultado mais eficiente é adotar metodologias de projeto mais modernas, mais ágeis e mais leves. Para resolver os problemas históricos, os gestores de projeto e os executivos têm de ser inovadores e criativos."

A área desenvolvimento de *software* da empresa escolhida para a realização deste estudo enfrenta alguns problemas, que a impedem de obter o maior sucesso possível na concretização dos múltiplos projetos em que está envolvida.

Sendo do interesse desta organização a reflexão sobre estes problemas, sobre os pontos a melhorar, e otimizar a forma como é feita a gestão de projetos nesta área, surgiu a possibilidade de realização de uma dissertação de mestrado aliada a um requisito de negócio real: melhorar a metodologia de gestão de projetos em uso na equipa de desenvolvimento de *software*.

# 2 Metodologia utilizada

## 2.1 Classificação da metodologia escolhida

Este trabalho irá realizar um estudo bibliográfico acerca das melhores práticas de gestão de projetos seguindo duas linhas de investigação: (1) Gestão através de metodologias tradicionais e (2) Gestão através de metodologias ágeis. Seguidamente irá efetuar um estudo de caso acerca do estado da gestão de projetos na direção de sistemas de informação da ZTelc, de forma a obter uma fotografia da situação atual, procurando depois soluções para os problemas que forem identificados e propondo uma melhoria generalizada ao processo de gestão de projetos nesta área.

Optou-se por um estudo deste tipo pois pretende-se conhecer de forma mais detalhada possível os processos atuais, bem como as dificuldades sentidas pelos diversos elementos da equipa. Runeson et al. (2009) refere que a metodologia de casos de estudo é adequada para muitos tipos de investigação em engenharia de *software*, e embora não permitam gerar os mesmos resultados do que experiencias controladas, permitem obter um entendimento mais aprofundado do fenómeno que está a ser estudado.

Segundo Runeson et al. (2009), um estudo de caso pode conter elementos provenientes de outros métodos de investigação, como um questionário ou entrevistas, e pode ser precedido por um estudo bibliográfico. O estudo de caso pode ainda utilizar análise de dados arquivados, entrevistas e observação da realidade como fonte de dados. Este trabalho irá ter em conta todos estes elementos.

Runeson et al. (2009) apresenta uma distinção entre 4 tipos de propósito de investigação:

1. Exploratória: pretende descobrir o que está a acontecer, procurando alcançar novos conhecimentos, permitindo a criação de novas ideias e hipóteses para novas investigações
2. Descritiva: tem por objetivo retratar um fenómeno ou situação específica
3. Explicativa: procura uma explicação para uma situação ou problema
4. De Melhoria: pretende melhorar um determinado aspeto do fenómeno estudado

O autor refere que os estudos de caso na disciplina de Engenharia de *Software* tendem a seguir um propósito de melhoria.

Runeson et al. (2009) faz ainda uma distinção entre os dois tipos de dados que podem resultar de um estudo de investigação: quantitativos e qualitativos, referindo que os casos de estudo se baseiam normalmente em dados qualitativos pois fornecem elementos descritivos mais aprofundados. No entanto, uma combinação de dados qualitativos com dados quantitativos pode proporcionar um melhor entendimento dos fenómenos estudados.
Os estudos de caso são ainda considerados flexíveis, no sentido em que os parâmetros chave do estudo podem ser alterados ao longo da sua vida.

O processo de investigação de um caso de estudo é constituído por 5 passos:

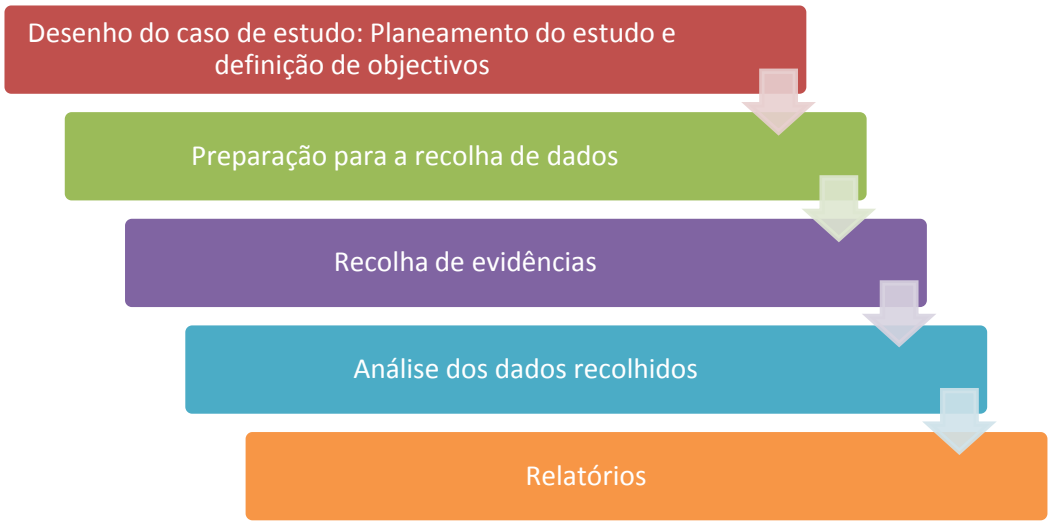



**Figura 3 - Processo de investigação de um caso de estudo baseado em Runeson et al. (2009)**

Runeson et al. (2009) refere que um estudo de caso nunca chega a conclusões com significado estatístico, mas permite interligar diferentes tipos de evidências, figuras, afirmações e documentos de forma a suportar uma conclusão relevante.

Para este trabalho a metodologia utilizada seguirá uma linha mais qualitativa mas com a análise de alguns dados quantitativos. Tem o propósito de investigação de melhoria e é complementada com entrevistas.

## 2.2 Entrevistas semiestruturadas

Runeson et al. (2009) indica que uma das principais metodologias de investigação relacionadas com o estudo de casos é a realização de questionários ou entrevistas.

Segundo Santos (2008), o guião utilizado para a realização de uma entrevista semiestruturada corresponde a uma lista de tópicos ou áreas gerais a cobrir, mas organizadas de uma forma que lhe permita a flexibilidade necessária para que durante a entrevista o investigador possa orientar a sua conduta de forma a obter o máximo de informação possível sobre o tema em estudo. Desta forma, neste tipo de entrevistas as questões podem não seguir a ordem prevista no guião, e podem inclusivamente ser colocadas questões que não se encontrem definidas inicialmente.
Santos (2008) apresenta as seguintes vantagens das entrevistas semiestruturadas:

- A possibilidade de acesso a muito mais informação
- A possibilidade de o investigador esclarecer de imediato alguns aspetos que surjam durante a entrevista
- Se realizadas na fase inicial do estudo, potenciam a geração de pontos de vista, orientações e hipóteses para o aprofundamento da investigação

Neste caso o mecanismo considerado mais adequado para avaliação e diagnóstico da situação atual e dos objetivos futuros para a gestão de projetos desta equipa foi a realização de um conjunto de entrevistas semiestruturadas aos diversos membros desta área (Anexo A). Estas conversas foram mantidas de forma regular durante todo o período de realização deste trabalho, de forma a garantir que se mantinha permanentemente alinhado com a realidade atual e com a realidade ideal desejada.

## 2.3 Revisão de literatura

A componente de investigação bibliográfica deste trabalho irá incidir particularmente sobre as seguintes questões:

- Planeamento do trabalho: estimativa de tempos e custos e planeamento das atividades de trabalho dos diversos elementos da equipa para a concretização do projeto.

- Monitorização e controlo: mecanismo de acompanhamento regular do estado do projeto para que seja possível o melhor acompanhamento do progresso e a identificação atempada de desvios.
- Relatórios e comunicação: mecanismos de comunicação entre o gestor e os membros da equipa do projeto e entre a equipa do projeto e a direção
- Métricas de desempenho em projetos de desenvolvimento de *software*: elementos que permitam aferir questões relacionadas com o progresso do projeto e com a qualidade do *software* que está a ser produzido.
- Gestão de equipas de desenvolvimento de *software*: considerações gerais sobre a especificidade da gestão de recursos humanos nos projetos de desenvolvimento de *software*, atendendo especificamente sobre questões relacionadas como a motivação, empenho e performance.
- Comunicação e informação: estudo dos mecanismos de informação e comunicação entre a equipa e os diversos *stakeholders*.

# 3 Revisão de literatura

## 3.1 As especificidades da gestão de projetos de *software*

Segundo Brooks (1995), a gestão de projetos de desenvolvimento de *software* é idêntica à gestão de qualquer outro tipo de projeto, mais do que os programadores acreditam, mas por outro lado, é bastante diferente, mais do que os gestores acreditam.

É importante compreender em que medida os projetos de desenvolvimento de *software* diferem de outros tipos de projetos, e que características tornam a sua gestão diferente, de forma a conseguir a sua gestão da forma mais eficiente possível.

Hughes e Cotterell (1999) identificam algumas características especiais dos produtos normalmente obtidos com os projetos de desenvolvimento de *software*:

- Invisibilidade: Noutros tipos de produtos, o progresso é muito evidente, mas nos projetos de *software*, com uma existência virtual, o progresso é muito mais difícil de medir.
- Complexidade: Por cada unidade monetária gasta, os produtos resultantes de um projeto de desenvolvimento de *software* possuem uma complexidade maior do que a resultante de outros produtos de engenharia.
- Flexibilidade: O *software*, não tendo existência material palpável é usualmente considerado pelas organizações como sendo o ponto mais simples para aplicação de qualquer alteração que ocorra na orla do negócio, o que condiciona estes tipo de projetos a um elevado nível de alterações

No artigo de Thamhain e Wilemon (1986) é referido um estudo em que vários gestores foram questionados sobre quais são os desafios mais frequentes na gestão de projetos de *software*. O resultado deste estudo é apresentado na seguinte tabela:

| Dificuldade | Respostas |
|---|---|
| Cumprir os prazos | 85% |
| Cumprir os constrangimentos do projeto | 83% |
| Comunicação eficaz entre os grupos de trabalho | 80% |
| Obter comprometimento por parte dos elementos das equipas | 74% |
| Estabelecer milestones que sejam mensuráveis | 70% |
| Gerir alterações | 60% |
| Chegar a acordo sobre o plano de projeto com os membros da equipa | 57% |
| Obter compromissos por parte da gestão | 45% |
| Gestão de fornecedores e sub contratos | 38% |

**Tabela 1 - Maiores desafios da gestão de projectos de software segundo Thamhain e Wilemon (1986)**

Hughes e Cotterell (1999) mostram um estudo baseado no resultado de um questionário (Thayer et.al.) que pretendeu identificar os principais problemas sentidos pelos gestores de projetos de desenvolvimento de *software*. Ao estudo deste questionário, os autores incluíram um estudo semelhante, mas com o objetivo de identificar os principais problemas sentidos pelos membros das equipas de projeto. O resultado destes estudos é apresentado na seguinte tabela:

| Problemas sentidos pelos gestores | Problemas sentidos pelos membros |
|---|---|
| Estimativas e planos pobres | Especificação inadequada do trabalho |
| Falta de padrões de qualidade | Ignorância técnica por parte dos elementos de gestão |
| Falta de medidas de controlo | Falta de padrões |
| Falta de apoio para tomadas de decisão organizacional | Falta de documentação atualizada |
| Falta de técnicas que tornem o progresso visível | Falta de comunicação |
| Indefinição de papéis. Quem faz o quê? | Alterações aos requisitos ou às plataformas tecnológicas de base |
| Critérios de definição de sucesso incorretos | Pressão exagerada para o cumprimento de prazos com impacto na qualidade |
| | Falta de controlo de qualidade |
| | Falta de formação |

**Tabela 2 - Problemas sentidos pelos gestores e pelos membros de projetos de *software* segundo Hughes e Cotterell (1999)**

Miguel (2010) identifica também algumas características que tornam a gestão de projetos de desenvolvimento de *software* diferente da gestão de outros tipos de projetos:

- Implicam uma mudança contínua
- São envolvidas pessoas de diferentes disciplinas
- A equipa muitas vezes trabalha em conjunto apenas num único projeto
- A produtividade é difícil de medir
- Os decisores estão, muitas vezes a trabalhar num domínio novo para eles
- As linhas de autoridade não estão muitas vezes claramente definidas
- Existem múltiplas visões do sucesso do projeto

A aplicação de modelos de gestão geral ao desenvolvimento de aplicações informáticas tem resultado recorrentemente em projetos concluídos fora dos constrangimentos inicialmente planeados. Miguel (2010) refere que "Se fosse efetuada uma auditoria pós morte a todos os projetos [de *software*], é muito provável que se encontrasse um tema recorrente – gestão deficiente".

## 3.2 Fatores críticos de sucesso para a gestão de projetos de *software*

É do conhecimento comum que a gestão de projetos de desenvolvimento de *software* apresenta resultados pouco satisfatórios, sendo habitualmente referido o Estudo do Standish Group, 2008 que refere que cerca de 20% dos projetos de *software* fracassam e cerca de 44% chegam ao fim com derrapagens sérias em termos de custos ou prazos, sendo estes valores superiores aos de outras especialidades de engenharia.

O artigo de Wateridge (1995) faz uma compilação de vários estudos sobre este tema, a partir de análises de vários autores e investigações próprias, dividindo os fatores críticos para a gestão de projetos de *software* em termos de Critérios de sucesso e fatores de sucesso:

**Critérios de sucesso**

O primeiro passo para aumentar o sucesso na gestão de projetos de desenvolvimento é garantir que existe um entendimento claro por parte de todos os membros da equipa acerca de

como será medido o sucesso. Noutras áreas, estes critérios de sucesso são facilmente identificáveis, no entanto nos projetos de *software* parece não haver um entendimento globalizado. O autor deste artigo realizou uma investigação no sentido de determinar estes critérios do sucesso para projetos de desenvolvimento de *software*. Para este efeito examinou mais de 100 projetos, realizando posteriormente entrevistas aos diversos elementos das equipas, onde se incluem gestores de projeto, sponsors, utilizadores, analistas de sistemas, equipas de suporte e outros, para tentar apurar as diversas visões acerca dos critérios de sucesso e quais os fatores que contribuíam para determinar o seu sucesso ou falha. Os resultados foram divididos entre projetos concluídos com sucesso, projetos falhados e todos os projetos. Desta forma é possível analisar como os critérios de sucesso utilizados definiram o resultado do projeto:

| Utilizadores | % | Gestores de Projeto | % |
|---|---|---|---|
| Atingimento dos requisitos do utilizador | 96 | Atingimento dos requisitos do utilizador | 86 |
| Satisfação dos utilizadores | 71 | Sucesso comercial | 71 |
| Cumprimento do orçamento | 71 | Cumprimento da qualidade | 67 |

**Tabela 3 - Três critérios de sucesso mais referidos para os projetos com sucesso. Estudo de Wateridge (1995)**

| Utilizadores | % | Gestores de Projeto | % |
|---|---|---|---|
| Atingimento dos requisitos do utilizador | 100 | Cumprimento do orçamento | 83 |
| Atingimento do objetivo | 100 | Cumprimento da calendarização | 78 |
| Satisfação dos utilizadores | 67 | Atingimento dos requisitos do utilizador | 78 |

**Tabela 4 - Três critérios de sucesso mais referidos para os projectos com falhados. Estudo de Wateridge (1995)**

| Utilizadores | % | Gestores de Projeto | % |
|---|---|---|---|
| Atingimento dos requisitos do utilizador | 96 | Atingimento dos requisitos do utilizador | 82 |
| Satisfação dos utilizadores | 71 | Cumprimento do orçamento | 72 |
| Cumprimento do orçamento | 67 | Cumprimento da calendarização | 69 |

**Tabela 5 - Três critérios de sucesso mais referidos para todos os projetos. Estudo de Wateridge (1995)**

Os resultados deste estudo evidenciam que não existe um entendimento comum aos vários perfis das esquipas do projeto em relação aos critérios de sucesso.

Em relação aos projetos falhados, verifica-se que a maior atenção dada pelos gestores se focou no atingimento de prazos e de orçamento, enquanto nos projetos com sucesso, a atenção dos gestores se focou no cumprimento dos requisitos dos utilizadores, no sucesso comercial do *software* desenvolvido e da sua qualidade.

Embora o autor referencie que por vezes o atingimento de prazos é extremamente importante (como a implementação de um sistema de faturação que tem de estar concluído no próximo ano fiscal), esta fixação dos gestores de projeto no calendário e nos orçamentos como principal critério de sucesso, em detrimento de critérios de qualidade ou de atingimento total das necessidades dos utilizadores é identificado como um dos grandes responsáveis pelo insucesso dos projetos de desenvolvimento de *software*.

Deste estudo conclui-se também que o entendimento generalizado por todos os elementos da equipa acerca dos critérios sucesso é difícil de atingir. Este entendimento pode mudar durante o curso do projeto, e muitos dos critérios são subjetivos. No entanto é entendido que os critérios de sucesso para projetos de desenvolvimento se devem basear na enumeração efetuada por Turner (1993) à qual deve ser acrescentado: "Atingimento de constrangimentos de qualidade", resultando assim na seguinte lista:

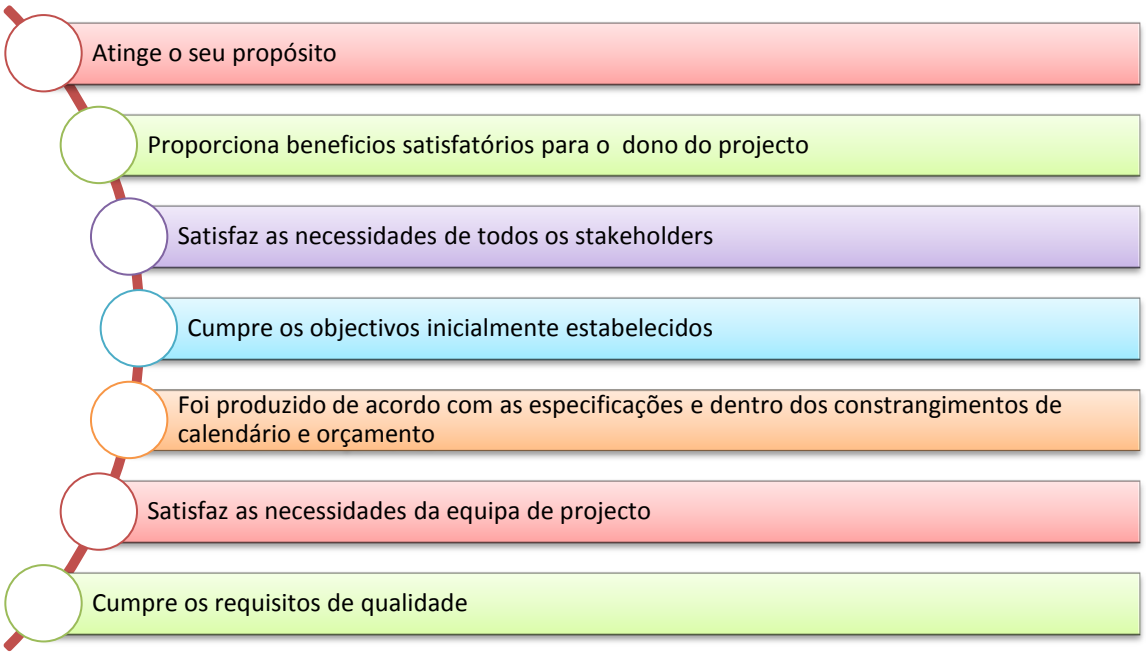



**Figura 4 - Critérios de sucesso na gestão de projetos de *software*. Baseado em Wateridge (1995)**

Wateridge (1995) frisa ainda que o gestor de projeto não deve ignorar as questões relacionadas com qualidade e mascarar a sua importância considerando-as parte do atingimento dos requisitos.

**Fatores de sucesso**

A segunda parte do artigo de Wateridge (1995) foca-se nos fatores de sucesso para a gestão de projetos de *software*, identificando que existe uma perceção diferente destes por parte dos membros da equipa e por parte dos gestores de projeto:

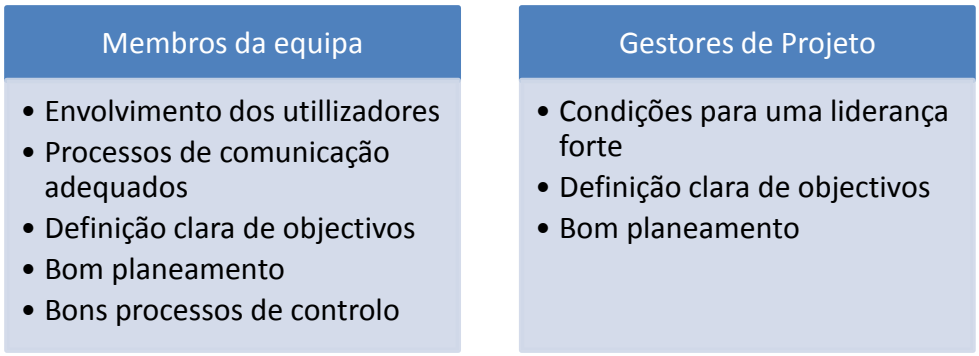



**Figura 5 – Fatores de sucesso segundo Wateridge (1995)**

Wateridge (1995) refere um inquérito realizado pela Management Consultancies Association (1993) que concluiu:

- O sucesso é normalmente conseguido através de equipas que trabalham para atingir objetivos previamente entendidos e acordados, seguindo um planeamento rigoroso e com um regime de controlo e gestão
- O insucesso é normalmente atribuível a objetivos pobres, confusos ou ambíguos, problemas de liderança, planeamento desadequado e falta de gestão

Miguel (2010) apresenta como fatores cruciais para a eficácia da gestão de projetos de *Software*, ressalvando que a ordem pela qual são apresentados não é arbitrária:

**As pessoas:** É fundamental que o gestor de projeto tenha plena consciência que "o trabalho de engenharia de *software* é um empreendimento humano intensivo". Neste contexto é

fundamental que os membros das equipas de projetos estejam não só devidamente motivados, como informados e que a sua participação seja o mais colaborativa possível, assegurando por um lado o seu sentimento de pertença ao projeto como também garantindo que os conhecimentos e experiência específicos de cada membro são absorvidos e integrados. Tal como refere Curtis et al. (2011) isto irá permitir "melhorar a eficiência das organizações de *software* no desenvolvimento de aplicações cada vez mais complexas, ajudando-as a atrair, desenvolver, motivar e reter o talento necessário para melhorar as suas capacidades de desenvolvimento de *software*"

**O Produto:** Por produto entende-se o resultado final da atividade de desenvolvimento que se pretende gerir. Antes que se possam realizar esforços para a correta gestão de um projeto de *software*, é essencial que sejam claramente entendidos e definidos os objetivos e âmbito do produto.

**O Processo:** "Um processo de *software* fornece a estrutura para o estabelecimento de um plano abrangente para o desenvolvimento. Existe um pequeno número de atividades que são aplicáveis a todos os projetos, independentemente da respetiva dimensão e complexidade. Um certo número de grupos de tarefas – marcos, produtos intermédios e pontos de garantia de qualidade – possibilitam a adaptação das atividades da estrutura às características do projeto de desenvolvimento e aos requisitos da equipa de projeto. No topo de tudo, existem atividades envolventes – garantia da qualidade, gestão da configuração e métricas, entre outras – que abrangem todo o modelo de processo, são independentes de qualquer atividade do modelo e ocorrem através de todo o processo."

**O Projeto:** Apenas com os pontos anteriores devidamente tratados se torna possível partir para o planeamento e gestão do projeto

Miguel (2010) faz uma referência a Boehm (1991)[1] que menciona "Os fatores críticos de sucesso dos projetos de *software* são os seguintes":

- Definição clara dos objetivos e limitações do projeto:
- Planeamento do projeto e organização do pessoal
- Envolvimento dos utilizadores
- Controlo do projeto
- Eficácia do gestor de projeto

[1] Boehm, B. /1991), Software Risk Management: Principles and Practices, IEEE Software, Vol.8, Nr. 1 (January 1991) pp.32-41

Especificamente em relação ao controlo, Miguel (2010) afirma: "Sem controlo o projeto vagueará sem objetivo. O controlo é, claramente muito importante e o nível a que é exercido deve ser alvo de cuidadoso planeamento. O controlo deve ser caracterizado por revisões frequentes ao longo do ciclo de vida do projeto, seguida das ações adequadas".

## 3.3 Modelos de gestão tradicional

### 3.3.1 Apresentação

Os modelos de gestão mais tradicionais, cujo corpo de conhecimentos se encontra extensivamente detalhado no PMBOK, um documento compilado pelo PMI (Project Management Institute) e que é o principal repositório de conhecimento acerca de gestão de projetos e que "inclui práticas tradicionais provadas que são amplamente aplicadas." (Miguel, 2010:33)

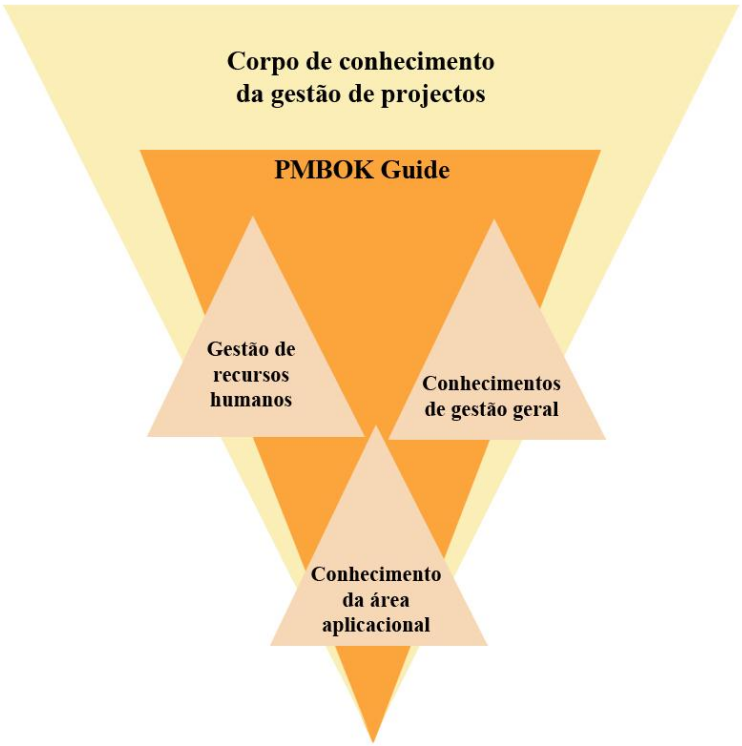



**Figura 6 - Relações entre as áreas de conhecimento da gestão de projectos. Miguel (2010)**

O PMBOK estrutura a gestão de projetos nos seguintes elementos:

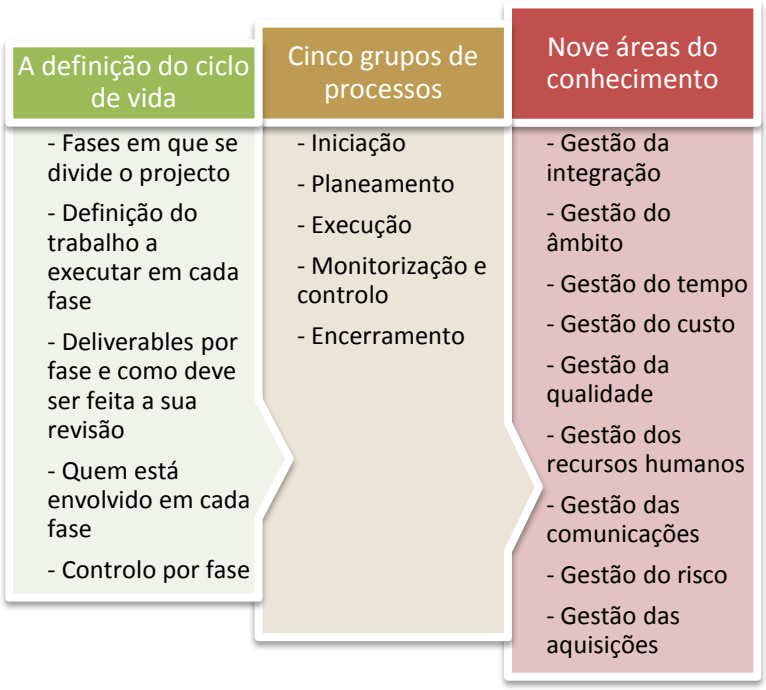



**Figura 7 - Estrutura de conhecimentos do PMBOK**

### 3.3.2 Ciclo de vida

O modelo de ciclo de vida tradicional na gestão de projetos de desenvolvimento de *software* segue uma linha sequencial de fases que se se vão sucedendo, começando uma quando termina a anterior. Este modelo é designado por modelo em cascata:

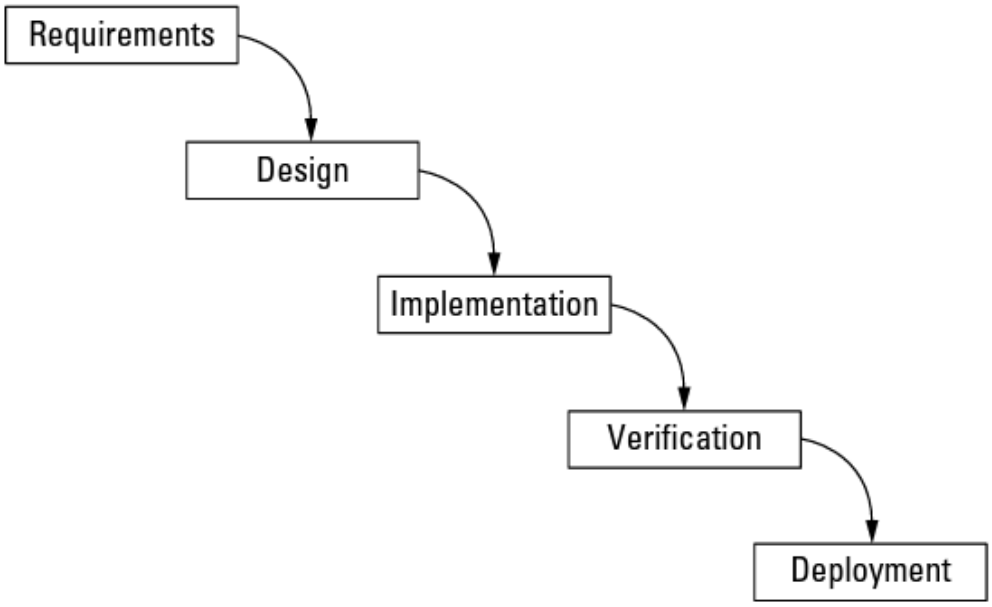



**Figura 8 - Modelo em cascata. Leffingwell (2011)**

Leffingwell (2011) refere que este modelo foi apresentado por Winston Royce em 1970, o qual reconheceu ser um modelo que não era adequado para todo o tipo de projetos e que não seria aplicável, por exemplo, a projetos grandes.

Larman (2008) descreve este modelo como iniciando o processo de produção com a fase de requisitos onde se tenta definir e descrever exaustivamente todos, ou praticamente todos, os requisitos antes de se iniciar o trabalho de desenvolvimento. O trabalho das fases seguintes é planeado em detalhe, normalmente através da representação das atividades em gráficos Gantt.
Com base nesta informação todas as equipas apresentam uma previsão de custos e de tempo para a execução de todas as tarefas até ao final do projeto.
Estas previsões são documentadas, revistas e aprovadas pelos *stakeholders* e apenas após esta aprovação as equipas iniciam os seus trabalhos.
A fase seguinte, onde é realizado o desenho e documentação do *software* deve ser executada antes de se iniciar o trabalho de implementação da fase posterior. Quando o trabalho de codificação está concluído, o *software* é passado para a equipa de testes, que o verifica exaustivamente. Uma vez concluído e aprovado este processo, a solução é disponibilizada aos clientes finais, entrando numa fase de operação e manutenção.
Cada fase é iniciada apenas com a receção de um entregável correspondente à conclusão da fase anterior.

Larman (2008) apresenta algumas vantagens e desvantagens deste modelo:

- Vantagens: todo o processo é lógico, estruturado e muito disciplinado. É facilmente entendido por todos os envolvidos e disponibiliza um conjunto de marcos (*milestones*) claros e tangíveis para todo o processo de desenvolvimento.
  O esforço inicial de planeamento e previsão de todos os requisitos do projeto e do seu desenho pode proporcionar uma grande economia de recursos nas fases mais avançadas.
  A atenção é dada à documentação em cada uma das fases com o objetivo de a fechar e de dotar os responsáveis pelas fases seguintes do conhecimento necessário para a continuidade do trabalho.
- Desvantagens: este modelo parte do princípio de que é possível prever no início do projeto todas as necessidades dos utilizadores, transformando-as em requisitos essencialmente imutáveis até final do projeto.

Uma vez que todo o projeto é pensado e desenhado antes da implementação, não é possível incorporar requisitos que sejam obtidos a partir da experiência de implementação.

### 3.3.3 Gestão de Recursos Humanos

O PMBOK define a equipa de gestão de projeto como um subconjunto da equipa de projeto, e que é responsável pela gestão de cada uma das fases. Para projetos pequenos esta responsabilidade pode ser dividida pelos diversos elementos da equipa, ou ser da responsabilidade de um elemento que o acumula com as restantes responsabilidades.

O gestor de projetos tem um conjunto de responsabilidades na gestão dos recursos humanos na equipa:

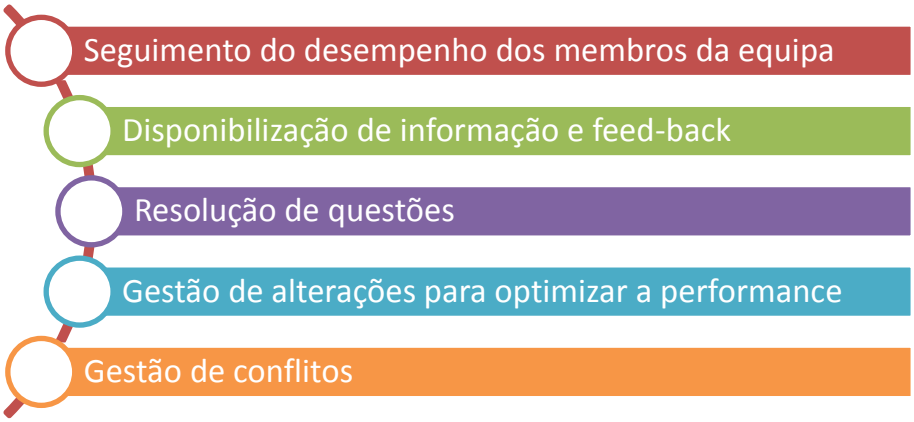



**Figura 9 - Responsabilidades do Gestor de Projetos. Baseado no PMBOK**

### 3.3.4 Planeamento do trabalho

O PMBOK refere que o planeamento consiste em:

- Estabelecer a abrangência total do projeto
- Definir e refinar os objetivos
- Planear o curso de ação necessário para o atingimento desses objetivos

O resultado do trabalho de planeamento é o plano de gestão do projeto.

Embora seja necessariamente assumido que o plano de projeto é sempre passível de alterações ao longo do ciclo de vida, a sua realização é fundamental para um entendimento adequado e abrangente, não só da solução final que se pretende implementar, como da equipa certa para a

sua concretização. Este documento permite ao gestor conduzir o curso do projeto, e funciona como uma ferramenta de apoio a decisões futuras.

Wysocki (2003) refere que é fundamental que exista uma compreensão por parte da gestão de que o plano é um documento dinâmico, e que é suposto que seja alterado ao longo do tempo de vida do projeto.

Wysocki (2003) define que um plano de projeto deve explicitar claramente:

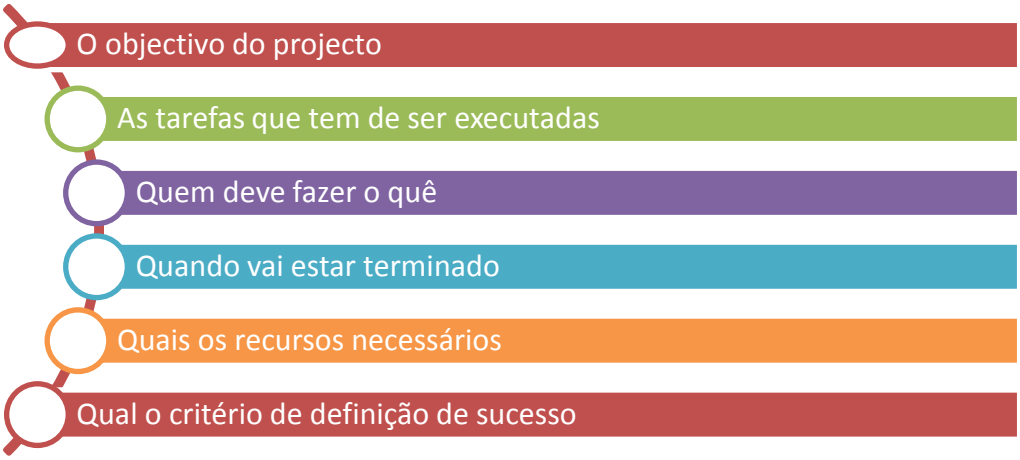



**Figura 10 - Componentes do Plano de Projeto segundo Wysocky (2003)**

A tabela seguinte apresenta os processos de planeamento segundo o PMBOK. Apenas estão representados os mais importantes para este trabalho. No entanto, a lista exaustiva está no Anexo D:

| Processo | Descrição e Observações |
|---|---|
| Recolher os requisitos | Definição e explicitação das necessidades dos *stakeholders* que devem ser endereçadas com este projeto. Segundo Miguel (2010) os requisitos devem cumprir um conjunto importante de critérios, antes de serem aceites para execução:<br>• Inequívocos (mensuráveis e testáveis)<br>• Auditáveis |

| Processo | Descrição e Observações |
|---|---|
| | • Completos<br>• Consistentes<br>• Aceitáveis<br>O PMBOK indica que que o sucesso do projeto está diretamente relacionado com o cuidado e atenção que é dada nesta fase à recolha e gestão dos requisitos do projeto e do produto a desenvolver. |
| Definir o âmbito | Descrição detalhada do projeto e do produto a obter com a sua execução. Miguel (2010) refere que esta descrição de âmbito é um trabalho dinâmico e que apenas estará completo na fase final do planeamento, quando são conhecidas todas as condições e limitações do projeto |
| Criar a WBS | Processo de subdivisão do trabalho do projeto em grupos mais pequenos e simples de gerir, e definição dos respetivos entregáveis |
| Definição de atividades | Definição concreta do conjunto de atividades necessárias para produzir cada um dos entregáveis do projeto |
| Estimação de tempos | Estimativa das unidades de tempo necessárias para a realização de cada uma das atividades com os recursos previamente estimados.<br>Tal como refere Brooks (1995), a maior causa de falha no cumprimento de projetos de desenvolvimento de *software* (maior que todas as outras juntas) é o cumprimento do prazo. Mas por que razão isto é tão comum?<br>Em primeiro lugar as técnicas de estimativa |

| Processo | Descrição e Observações |
|---|---|
| | estão pouco desenvolvidas, e partem de uma assunção irrealista de que no final tudo vai correr bem.<br>Em segundo lugar, as técnicas de estimativa falaciosamente confundem esforço com progresso. |
| Desenvolver o cronograma | Criação de um cronograma a partir da análise da sequência de atividades, durações, necessidades de recursos e restrições de calendarização |
| Planeamento de comunicações | Determinação das necessidades de informação dos *stakeholders* e definição de um sistema de comunicação |

**Tabela 6 - Processos da fase de planeamento segundo o PMBOK**

Wysocki (2003) reforça a necessidade de um planeamento cuidado com a utilização da seguinte imagem:

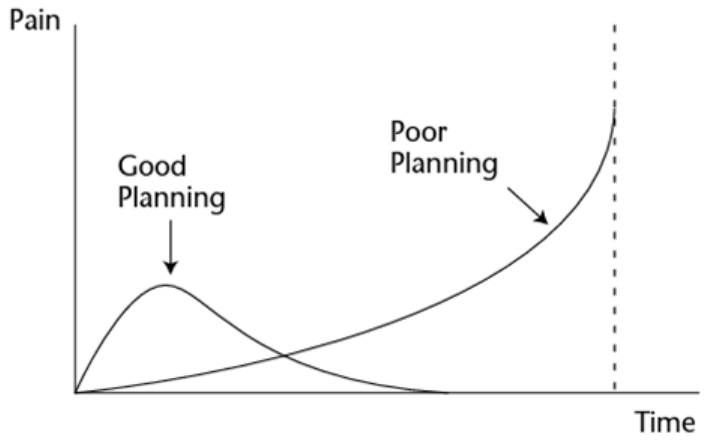



**Figura 11 - Efeitos do planeamento ao longo do tempo para Wysocki (2003)**

#### 3.3.4.1 Planeamento do trabalho em projetos pequenos

O conjunto de processos definidos no ponto anterior pode constituir um esforço exagerado na gestão de projetos de pequena dimensão. Miguel (2010) refere que neste tipo de projetos podem ser utilizadas ferramentas e técnicas como *Mind Mapping*, que permitem obter uma lista de atividades de uma forma mais simples, não sequencial. O mesmo autor diz ainda "Pequenos projetos, embora exijam igualmente planeamento, devem ser abordados com sensatez no que se refere à complexidade dos detalhes desse planeamento"

### 3.3.5 Execução do trabalho

Esta fase é constituída pela coordenação de pessoas e outros recursos para além da realização das tarefas que compõem o plano. A tabela seguinte apresenta os processos definidos pelo PMBOK mais relevantes para este estudo (a listagem completa é apresentada no Anexo E)

| Processo | Descrição e Observações |
|---|---|
| Gerir e dirigir a execução do projeto | Realização do trabalho definido no plano de projeto de forma a permitir o atingimento dos objetivos do projeto |
| Desenvolver a equipa do projeto | Melhorar as competências da equipa, da sua interação e de toda a envolvência da equipa de forma a melhorar a sua performance no projeto |
| Distribuição de informação | Garantir que a informação relevante chega aos *stakeholders* |
| Gerir as expectativas dos *stakeholders* | Comunicar e trabalhar com os *stakeholders* para garantir que são atingidos os seus objetivos e que são de imediato tratadas quaisquer questões |

**Tabela 7 - Processos da fase de execução segundo o PMBOK**

### 3.3.6 Monitorização e controlo

Permite acompanhar, medir e avaliar regularmente e de forma consistente variações em relação ao plano do projeto. A tabela seguinte apresenta os processos definidos pelo PMBOK mais relevantes para este estudo (a listagem completa é apresentada no Anexo F)

| Processo | Descrição e Observações |
|---|---|
| Monitorizar e controlar o trabalho do projeto | Revisão e regulação do progresso do trabalho que está ser realizado para que sejam atingidos os objetivos definidos no plano de projeto. A monitorização inclui relatórios de ponto de situação, medidas de progresso e previsões de evolução. Os relatórios de performance possuem informação relacionada com o âmbito, calendarização, custos, recursos, qualidade e riscos. |
| Realizar o controlo de alterações | Revisão de todos os pedidos de alteração, aprovação e gestão de alterações no plano e nos entregáveis do projeto |
| Verificar o âmbito do projeto | Formalização de aceitação dos entregáveis do projeto que foram concluídos |
| Controlar o âmbito | Monitorização do estado do âmbito do projeto e do produto e gestão das alterações à *baseline* de âmbito |
| Controlar a calendarização | Monitorização do estado do projeto e atualização do seu progresso. Gestão de alterações à *baseline* de calendarização |
| Reportar a performance | Recolher e distribuir informação de performance incluindo relatórios de estado, medidas de progresso e previsões acerca da evolução do projeto |

**Tabela 8 - Processos da fase de monitorização e controlo segundo o PMBOK**

### 3.3.7 Relatórios e comunicação

Segundo o PMBOK, a gestão de comunicação do projeto inclui os processos necessários para garantir que se consegue de forma atempada os elementos para a recolha, distribuição, armazenamento e disponibilização de informação do projeto.

Uma vez que os gestores de projeto passam grande parte do seu tempo a comunicar com os membros da equipa e com os outros *stakeholders*, uma comunicação efetiva e estruturada é essencial para assegurar o cumprimento de todas as expectativas.

Miguel (2010) afirma que, para que o gestor de projeto possa garantir que as atividades estão a ser realizadas de acordo com o plano, é necessário um conjunto de relatórios de progresso. Este conjunto de relatórios deve possuir as seguintes características:

- Proporcionar informação de situação oportuna, completa e precisa
- Não adicionar muito trabalho burocrático
- Estar rapidamente acessível à equipa de projeto e à gestão de topo
- Ser facilmente compreensível para todos aqueles que têm uma necessidade de conhecer a situação do projeto

Miguel (2010) apresenta um conjunto de tipos de relatórios de progresso:

| Tipo de relatório | Descrição |
|---|---|
| Relatórios do período corrente | Respeitam o período mais recente e refletem o trabalho realizado. Deve dar foco especial às atividades concluídas e aos desvios entre datas de conclusão reais e planeadas. Caso existam atividades que não tenham evoluído de acordo com o plano, este relatório deve incluir uma justificação e a apresentação das medidas corretivas que serão implementadas. Este relatório deverá ser realizado em duas versões: a primeira é realizada pela equipa do projeto para o gestor de projeto, e deverá ter uma periodicidade semanal.<br>A segunda é realizada pelo gestor de projeto para a gestão de topo e poderá ter uma |

| Tipo de relatório | Descrição |
|---|---|
| | periodicidade quinzenal ou mensal |
| Relatórios cumulativos | Contêm a história do projeto para todo o período. São mais informativos que o relatório do período corrente, pois mostram tendências no progresso do projeto. Estes relatórios podem ser realizados ao nível da atividade ou ao nível do projeto |
| Relatórios de exceção | Apresentam os desvios ao plano e destinam-se tipicamente à gestão de topo, e devem ser desenhados de forma a permitirem uma leitura simples e uma interpretação rápida |

**Tabela 9 - Tipos de relatórios para Miguel (2010)**

## 3.4 Modelos de gestão ágeis

Nos últimos anos tem surgido um conjunto de modelos alternativos para a gestão de projetos de *software*, tendo por base a posição de que a gestão tradicional aplicada a este tipo de projetos estava a ter resultados menos positivos do que noutras áreas. Estas metodologias são conhecidas de forma global como modelos ágeis.

No entanto, Leffingwell (2011) refere que muitas das práticas disponibilizadas pelas metodologias tradicionais continuam a oferecer conhecimentos e ferramentas de valor efetivo e comprovado pela experiência de muitos anos de utilização, pelo que muitas delas deverão continuar a ser utilizadas. No entanto, e segundo o mesmo autor, é necessário considerar o trabalho que está a ser realizado pela comunidade que está a desenvolver estas metodologias ágeis e que estas têm apresentado resultados surpreendentes.

As metodologias ágeis decompõem os grandes degraus do modelo de ciclo de vida "em cascata" num conjunto de interações autocontidas em unidades de tempo. Estas unidades mais pequenas aceleram dramaticamente o feedback, produzindo grandes benefícios para o ciclo de vida do projeto.

### 3.4.1 Apresentação

Segundo Leffingwell (2011), em 2001 os responsáveis por várias metodologias de desenvolvimento ágil criaram o manifesto ágil. Este grupo de pessoas partiu do pressuposto de que face a todas as dificuldades que se conheciam na gestão de projetos de *software*, devia existir uma forma melhor de os gerir, e definiu um conjunto de princípios nucleares que definem este tipo de metodologias:

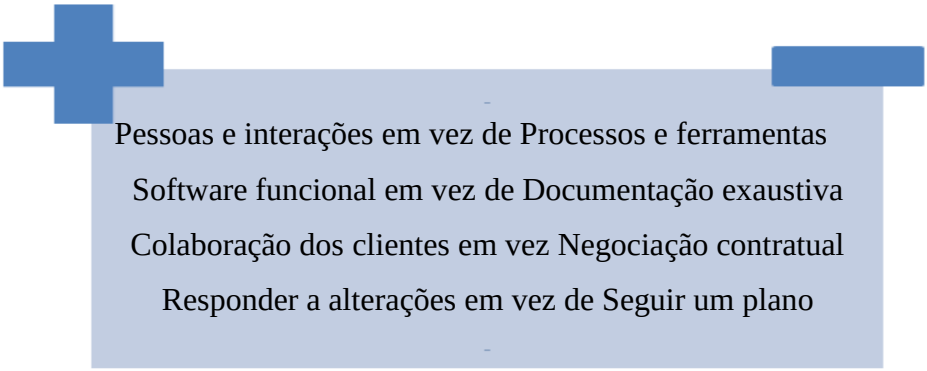



**Figura 12 - Princípios fundamentais das metodologias ágeis segundo Leffingwell (2011)**

Estes princípios não anulam os valores representados à direita, apenas atribuem um peso maior aos representados à esquerda.

Cohn (2005) refere que as equipas ágeis valorizam as pessoas e as suas interações, em vez dos processos e ferramentas, pois baseiam-se no princípio de que uma equipa que funcione bem e seja composta por bons profissionais, mas com ferramentas medíocres, irá sempre superar uma equipa disfuncional com elementos menos bons mas com excelentes processos e ferramentas.

As equipas ágeis valorizam *software* funcional à documentação exaustiva, pois isso conduz incrementalmente a uma versão mais estável e melhorada do produto que está a ser desenvolvido.

As equipas ágeis valorizam a resposta à mudança ao seguimento de planos, pois o seu objetivo principal é criar o maior valor possível para os clientes e utilizadores do projeto.

Para uma equipa ágil, um plano é uma visualização do futuro, mas muitas visualizações são possíveis, e à medida que a equipa vai ganhando conhecimento e experiência, isto vai se refletindo nos planos.

Têm proliferado uma grande quantidade de metodologias alternativas, embora baseadas no mesmo manifesto e nos mesmos princípios basilares. Leffingwell (2011) apresenta o resultado de um inquérito que mostra a relação de utilização, permitindo identificar as mais utilizadas:

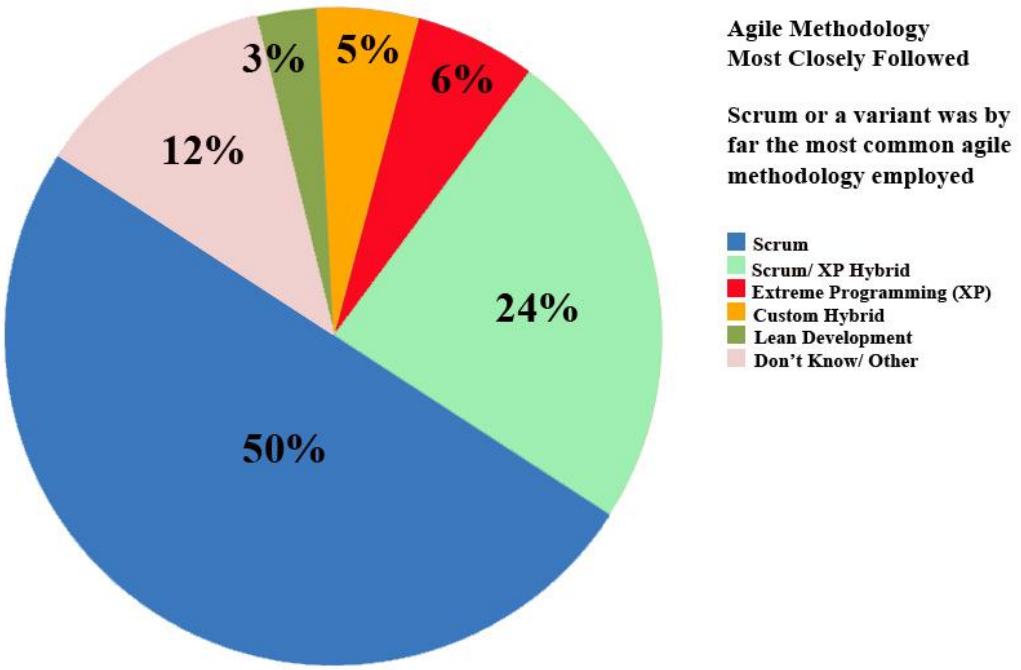



**Figura 13 - Comparação de uso de metodologias ágeis segundo Leffingwell (2011)**

O resultado deste questionário reflete que a metodologia mais utilizada é o *Scrum*, com 74% de utilização (em 24% dos casos em conjunto com a XP).
Este trabalho irá assumir as metodologias ágeis de forma geral embora siga as definições do *Scrum* onde existirem diferentes interpretações do modelo ágil.

Cohn (2005) refere que os princípios basilares dos movimentos ágeis levam a processos de desenvolvimento de *software* que são altamente iterativos e incrementais e que permitem produzir *software* funcional e testado no final de cada iteração. As equipas funcionam de acordo com os seguintes princípios:

- O trabalho é feito em nome de uma só equipa
- O trabalho é realizado em iterações curtas
- É disponibilizado *software* testado e funcional ao fim de cada iteração
- É dada atenção especial às prioridades do negócio
- As equipas adaptam-se

#### 3.4.1.1 Vantagens das metodologias ágeis

Rasmusson (2010) apresenta uma versão de alto nível das grandes vantagens de uma metodologia ágil. Com este tipo metodologia é disponibilizado algo de valor para o cliente a cada semana (ou outro período definido como tempo para o "sprint"). Em vez de receber apenas um conjunto de documentos, planos e relatórios, o cliente pode receber de forma regular uma solução de *software* funcional e devidamente testada. Para além das vantagens imediatas, esta situação oferece um conjunto de vantagens adicionais:

1. Os problemas grandes são divididos em problemas mais pequenos, mais simples e mais fáceis de gerir.
2. O cliente sabe que a equipa está a dar prioridade aos assuntos realmente importantes, e não a questões que não se traduzem em valor real. A documentação e planos não são desprezados, mas têm sempre por base elementos de *software* funcionais.
3. A necessidade de entregar um componente de *software* totalmente funcional no final de cada ciclo cria uma grande pressão para que sejam realizados muitos testes, garantindo que o projeto final terá um nível de qualidade superior.

4. Acompanhamento de feedback regular. A cada ciclo o cliente avalia e redefine o rumo que o projeto está a seguir.
5. O caminho pode ser corrigido. Isto faz com que o destino final seja mesmo o destino e não a imagem que se tinha do destino. Os planos mudam, a importância dos projetos muda e o cliente nem sabe bem quais são as suas verdadeiras necessidades. A possibilidade de recolher feedback e proceder às correções necessárias a cada iteração garante que a solução implementada será a que o cliente precisa e a que trará mais valor para a organização.

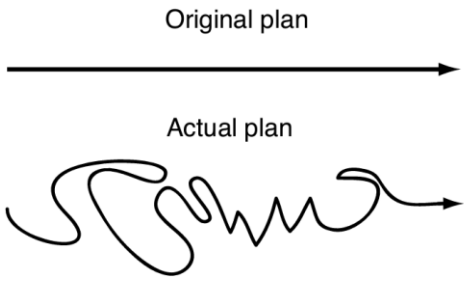



**Figura 14 - O verdadeiro caminho do plano para Rasmusson (2010)**

6. Cada elemento da equipa passa a ser responsável e responsabilizado. Cada membro assume o compromisso de atingir um determinado conjunto de objetivos em cada ciclo, e isso torna-o responsável por gerir o seu calendário, as suas expectativas e a qualidade do componente que está a desenvolver. Esta forma de trabalhar coloca muita responsabilidade sobre cada membro da equipa, mas por outro lado confere-lhe muito mais sentido de pertença e responsabilidade ao projeto.

### 3.4.2 Ciclo de vida

Para Leffingwell (2011) o surgimento de metodologias mais simples, leves e ágeis é o evento mais significativo que pode afetar as empresas que envolvem desenvolvimento de *software* desde a apresentação do modelo em cascata em 1970.

Para além das desvantagens do modelo em cascata (já identificadas no ponto 3.3.2) Leffingwell (2011) indica que uma das grandes causas para as dificuldades em obter sucesso na gestão de projetos de desenvolvimento de *software* se deve à rigidez do processo de definição de requisitos deste modelo. O autor refere um estudo realizado com uma amostra de 1.027 projetos de IT no Reino Unido onde foi concluído que a definição de todos os requisitos numa fase inicial seguida de um grande intervalo de tempo até que os resultados sejam entregues já não é apropriada. O elevado nível de alterações ao nível das necessidades de negócio sugere que qualquer assunção de que irão existir poucas alterações aos requisitos após a documentação inicial está fundamentalmente incorreta.

Para este autor, o modelo em cascata traz consigo uma armadilha, designada habitualmente por triângulo de ferro:

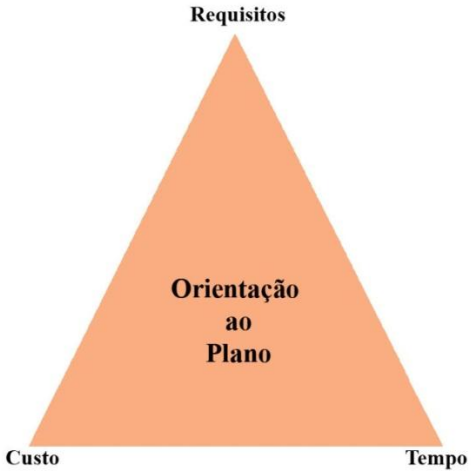



**Figura 15 - A armadilha do triângulo de ferro no modelo em cascata, segundo Leffingwell (2011)**

Uma vez que o custo, o tempo e os requisitos são fixos, a única variável flexível é a qualidade, com os resultados desastrosos que isso pode trazer para o projeto.

O ciclo de vida proposto na gestão ágil propõe uma abordagem diferente, em vez de se partir para um processo de definição e planeamento integral no início, é seguida uma abordagem mais de exploração, constituindo um modelo iterativo, em que a documentação é muito simplificada e em que se seguem modelos como documentos de visão ou modelos de utilização, que são utilizados como definição inicial do que se pretende desenvolver. Com base nestas definições iniciais, o processo iterativo irá permitir chegar às verdadeiras necessidades e requisitos.

Rasmusson (2010) refere que no ciclo de vida dos projetos ágeis, o trabalho de análise, programação, *design* e testes não tem propriamente um fim estas atividades são repetidas em cada iteração (sprint).

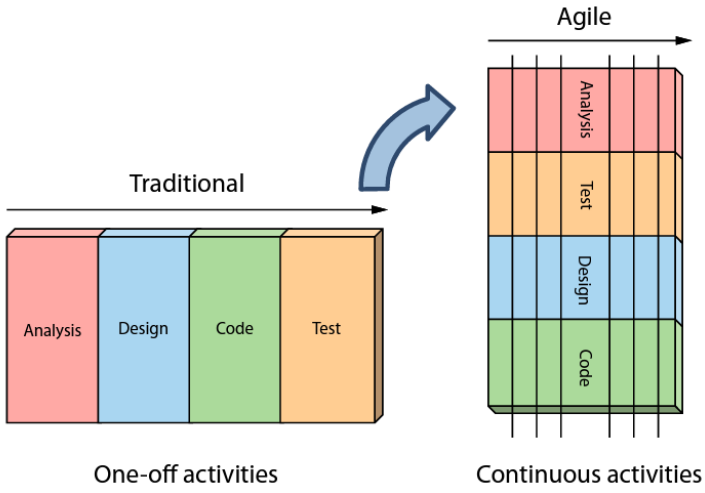



**Figura 16 - Ciclo de vida da gestão ágil. Fonte:Rasmusson (2010)**

Esta forma de trabalho conduz a níveis de eficiência maiores, uma vez que cada uma das fases não pode ser tomada de forma isolada, levando a uma maior proximidade entre os diferentes membros da equipa do projeto.

### 3.4.3 Gestão de Recursos Humanos

Schwaber & Sutherland (2010) referem que equipas ágeis são desenhadas para otimizar a flexibilidade e a produtividade. Para este efeito estas equipas são auto-organizadas, multifuncionais e trabalham em iterações.

Rasmusson (2010) compara uma equipa de projeto ágil com uma mini "*startup*", em que as pessoas pegam e fazem tudo o que podem para que o projeto tenha sucesso, independentemente do seu título ou função.

Isto não significa que cada membro não tenha as suas competências técnicas e especialidades, e que normalmente se dediquem às áreas que dominam, mas não existe uma especialização ao nível de analista, programador ou *tester*. Todos os membros da equipa realizam as tarefas correspondentes à necessidade da situação.

Para Rasmusson (2010), uma das melhores consequências da forma de trabalho ágil ao nível das equipas é a forma como todos se assumem como responsáveis e responsabilizáveis pelo projeto. O sucesso depende unicamente do esforço de equipa, e não de pessoas isoladas a trabalhar de forma independente uns dos outros.

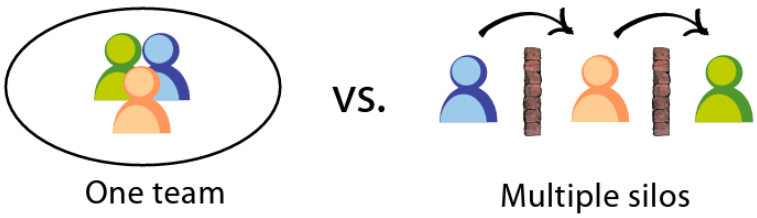



**Figura 17 - Impacto das metodologias ágeis na equipa. Fonte: Rasmusson (2010)**

Este autor define ainda as equipas ágeis como tendo as seguintes características:

- Auto organizadas: A equipa de forma coletiva e autónoma, sem que exista uma hierarquia de controlo ou poder, define a forma como vai chegar a um determinado objetivo técnico. Para que a equipa consiga atingir este nível de auto organização, a direção tem de ter abertura para que:
    - Seja a equipa a definir as estimativas e seja em conjunto a responsável pelo projeto.

    - As pessoas da equipa possam ter iniciativa e não estejam dependentes de uma figura de autoridade que vá ditando o destino e dando ordens aos restantes membros.
- Com elevado nível de responsabilidade, poder e responsabilizáveis: estas equipas assumem a responsabilidade de entregar ao cliente aquilo a que se comprometeram e aceitam ser responsabilizadas caso isso não aconteça. Mas isto só é possível se as equipas tiverem realmente poder e autonomia. Os membros da equipa têm de poder tomar as suas decisões e resolver os seus problemas sem depender da autorização de uma figura hierárquica. Isto pode levar a erros ocasionais, mas isso deve ser aceite como parte do processo.
- Multifuncional: A equipa deve ter pessoas com um leque de competências o mais abrangente possível para que seja possível disponibilizar ao cliente qualquer funcionalidade que este solicite. Uma equipa ágil funciona melhor com generalistas do que com especialistas, embora possa recorrer a estes em situações específicas. Mas a capacidade de adaptação do generalista é mais importante.

Schwaber & Sutherland (2010) salientam a importância da multidisciplinaridade de toda a equipa, indicando que todos devem fazer tudo, mesmo que para isso tenham de aprender novas tecnologias ou técnicas. Os autores salientam também a importância da inexistência de títulos hierárquicos na equipa, e de que esta é uma regra primordial que não aceita exceções. As equipas ágeis também não permitem a existência de subequipas especializadas em domínios particulares. Os autores referem ainda que dimensão ideal de uma equipa ágil é entre 5 e 9 elementos (excluindo o *Product Owner* e o *Scrum Master*)

Para além da equipa, o Scrum (e muitas das metodologias ágeis) definem apenas duas funções adicionais (Rasmusson, 2010):

- *Product Owner*: Esta função tem a responsabilidade de manter a ponte com o cliente final, sendo responsável por receber todos os requisitos e definir as suas prioridades. Normalmente as prioridades são definidas do ponto de vista do negócio, sendo envolvida a equipa do projeto para a abranger no projeto e para validar elementos técnicos que as possam condicionar. Para Cohn (2005), um dos principais papéis do *Product Owner* é assegurar que todos os membros da equipa possuem a mesma visão

para o projeto, e gerir as prioridades de forma a garantir que o trabalho está sempre a ser realizado sobre as funcionalidades mais importantes para o cliente e tomando decisões que conduzam a um bom retorno de investimento. Schwaber & Sutherland (2010) referem que o *Product Owner* é o único responsável pela gestão do *product backlog* [2] e pela garantia do ganho de valor para a empresa obtido com o trabalho que a equipa realiza. É também o responsável por garantir que esta informação está visível para os membros da equipa. Os autores referem ainda que este perfil deve ser atribuído a uma só pessoa por cada produto, e que a organização tem de respeitar as suas decisões, permitindo que só esta pessoa possa definir as prioridades que irão conduzir o trabalho da equipa. Por seu turno, a equipa não pode receber definições de prioridades de mais ninguém a não ser do *product owner*. As suas decisões estão visíveis no conteúdo e priorização do *product backlog*. Esta visibilidade coloca o *product owner* numa posição muito exigente mas ao mesmo tempo muito desafiante.

- *Scrum Master*: Este perfil não é uma figura de autoridade, mas sim um elemento de apoio e suporte para a equipa seguir o caminho ágil, fornecendo informação e promovendo os princípios e filosofias ágeis. Schwaber & Sutherland (2010) referem que o *Scrum master* deve ensinar os princípios ágeis à equipa, apoiando-a e liderando-a de forma a tornar-se mais produtiva e produzindo produtos de maior qualidade. Deve também ajudar a equipa a fazer o seu melhor num ambiente organizacional que pode não estar optimizador para a realização projetos de desenvolvimento complexos.

### 3.4.4 Planeamento do trabalho

Cohn (2005) refere que as equipas ágeis realizam planeamento pelo menos a 3 níveis distintos: a *release*, a iteração e o dia corrente. No entanto, este planeamento pode ser estendido a níveis mais elevados ou a horizontes mais distantes:

[2] Repositório de funcionalidades a realizar numa release

**Figura 18 - Horizontes de planeamento ágil. Baseado em Cohn (2005)**

### 3.4.4.1 User Stories

Segundo Cohn (2005), as equipas ágeis têm por objetivo completar e disponibilizar funcionalidades de valor para o cliente em vez de completarem tarefas isoladas (que eventualmente se iriam combinar numa funcionalidade). Para atingir este fim são utilizadas *user stories*, que são uma forma muito leve e prática de expressar requisitos. Uma *user story* é uma descrição breve de uma funcionalidade tal como esta é vista pelo utilizador. A forma de expressar estas histórias não tem um formato rígido, mas uma forma normalmente utilizada é:

como um <tipo de utilizador>, eu quero <capacidade> para que <valor de negócio>

- Exemplo: como um comprador de livros, eu quero procurar um livro por ISBN para que possa encontrar o livro certo mais rapidamente

**Figura 19 - Formulação de user stories. Adaptado de Cohn (2005)**

O trabalho para obter e documentar cada *user story* não é totalmente realizado no início. Em vez de se elaborar uma longa lista de requisitos, é criada apenas uma pequena descrição, que é normalmente mantida em cartões ou num documento, e que serve de base para conversas entre o *product owner* e os membros da equipa. Estas conversas devem ocorrer tantas vezes

quanto necessário e devem incluir quem quer que seja necessário. A documentação escrita pode continuar a existir quando os requisitos seguem uma abordagem deste tipo, mas o *focus* é alterado de forma dramática da componente escrita para a componente verbal

#### 3.4.4.2 Planeamento da *Release* (*Release Planning*)

Cohn (2005) refere que o planeamento de uma *release* é realizado através de um conjunto de *user stories* que devem ser implementas em conjunto. Este planeamento é realizado no início do projeto, mas não é um esforço isolado, sendo atualizado no início de cada iteração com o conhecimento que foi adquirido na iteração anterior e sofre a adaptação necessária de forma a refletir sempre as expectativas acerca do que vai ser incluído na *release*.

O nível de gestão ao nível da *release* permite obter um grande foco nas prioridades de negócio.

Do planeamento de *release* resulta o *Product Backlog*, que consiste numa lista priorizada de tudo o que será necessário para atingir os objetivos definidos para a *release* (Schwaber & Sutherland, 2010).

#### 3.4.4.3 Planeamento de iteração (Sprint Planning)

É realizado no início de cada iteração, e tem por base o trabalho desenvolvido na iteração anterior. O *product owner* identifica as funcionalidades mais importantes que a equipa deve endereçar na próxima iteração. Neste planeamento o horizonte é mais curto que no planeamento da *release* e os componentes do plano devem ser mais pequenos. A atenção é dada às tarefas que serão necessárias para transformar um pedido de funcionalidade num elemento de *software* funcional e testado (Cohn, 2005).

Do resultado do planeamento de iteração (sprint) irá resultar o *sprint backlog*, uma lista de tarefas necessárias para avançar nos objetivos do *product backlog*, que vão sendo concretizados a cada iteração (Schwaber & Sutherland, 2010).

#### 3.4.4.4 Planeamento diário (Daily scrum meeting)

Este planeamento é realizado através de uma reunião muito rápida, designada usualmente por "daily stand-up meeting" onde se coordena e sincroniza os esforços diários (Cohn, 2005).

Schwaber & Sutherland (2010) referem que durante esta reunião cada membro da equipa deve responder a 3 questões:

1. O que realizou deste a ultima reunião?
2. O que vai fazer antes da próxima reunião?
3. Que obstáculos encontrou e estão a impedir o progresso?

#### 3.4.4.5 Planeamento do produto

Cohn (2005) refere que o planeamento de produto é realizado pelo *product owner*, procurando objetivos mais distantes do que a *release* atual, focando a atenção no planeamento da evolução do produto ou sistema.

#### 3.4.4.6 Planeamento de portfólio

Cohn (2005) refere que o planeamento de portfólio envolve a seleção de produtos que melhor irão implementar a visão estabelecida pelo planeamento estratégico da organização.

#### 3.4.4.7 Condições de satisfação

Segundo Cohn (2005), os planeamentos realizados para a *release* e para a iteração devem satisfazer um conjunto de objetivos, entre os quais estarão necessariamente os de calendarização, âmbito, orçamento e qualidade. Estas são as condições de satisfação do *product owner*.

Quando não é possível atingir estas condições de satisfação, estas têm de ser ajustadas, o que leva a que o planeamento da *release* e a exploração das condições de satisfação do product owner seja um processo muito iterativo, tal como representado na figura seguinte:

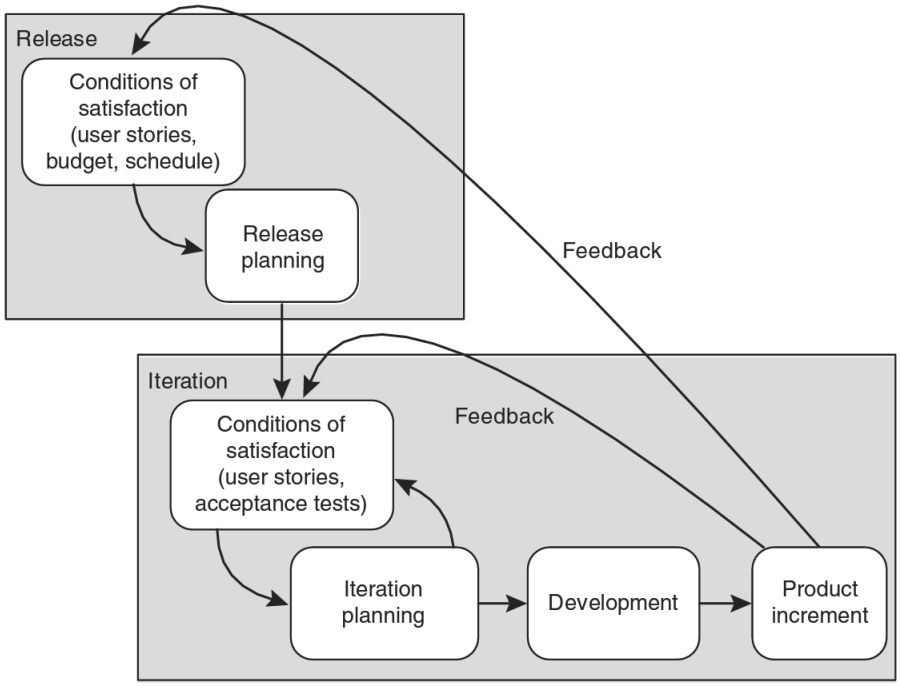



**Figura 20 - Condições de satisfação conduzem o planeamento de *releases* e iterações. Fonte: Cohn (2005)**

O plano de *release* é estabelecido para um período aproximado de 1 a 6 meses. Este plano é posteriormente utilizado como base para o planeamento da primeira iteração, que tal como o planeamento de *release*, considera as condições de satisfação do *product owner*.

Para uma iteração, as condições de satisfação são normalmente as funcionalidades que se pretende implementar.

### 3.4.4.8 Estimativas de tempo em metodologias ágeis

Segundo Cohn (2005), as metodologias ágeis diferenciam a estimativa de duração da estimativa de tamanho. Em primeiro lugar é estimado o volume de esforço necessário para realizar a tarefa e depois, com base no volume definido, é calculada duração necessária para o realizar. Este processo permite obter estimativas muito mais rigorosas do que simplesmente tentar adivinhar a duração apenas pela ideia que temos do tempo que a tarefa pode levar.

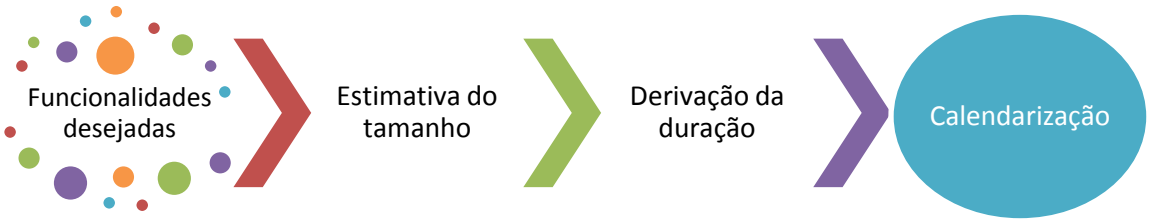



**Figura 21 - Processo de estimação de tempos em metodologias ágeis, segundo Cohn (2005)**

Cohn (2005) indica que nas metodologias ágeis a estimativa do tamanho pode ser realizada de duas formas:

1. *Story Points*: São unidades de medida para expressar de forma relativa o tamanho de uma *user story* em relação a outras. Os valores não são importantes, o que é importante é a diferença entre o peso do esforço (por exemplo uma *user story* com 2 pontos deve ter o dobro do esforço de uma com 1). O autor recomenda que se escolha a história que pareça ter o tamanho médio e atribuir-lhe o valor 5, a partir daí, e por comparação, atribuir às restantes tamanhos entre 1 e 10. Num projeto de gestão ágil, não é incomum começar uma iteração com requisitos incompletos. No entanto, todas as *user stories* têm que ter uma estimativa de tamanho. Estes pontos são fundamentais para o cálculo da velocidade.

   O somatório das estimativas de todos os pontos das funcionalidades desejadas permite obter o tamanho do projeto (*release*). A divisão do total de pontos de projeto pelo valor da velocidade da equipa permite calcular quantas iterações serão necessárias para a sua implementação. Sabendo este número e sendo fixa a duração das iterações, é possível estabelecer uma calendarização.
2. *Ideal Time*: O tempo ideal corresponde ao tempo que se demoraria a realizar uma tarefa se essa fosse a única coisa a fazer. Este tempo é diferente do real, pois é impossível dedicar todo o tempo a uma tarefa. Ao estimar desta forma deve considerar-se que (1) a história que está a ser estimada é a única tarefa em mãos (2) tudo o que for necessário para a realização do trabalho estará pronto antes deste começar (3) não irão existir quaisquer interrupções. Desta forma, o tempo ideal é apenas uma medida relativa de estimativa de tamanho, idêntico aos *story points*. A duração verdadeira deve ser calculada utilizando a velocidade tal como no exemplo

anterior. Esta opção encerra o perigo de confusão entre o tempo ideal e o tempo de calendário, sendo a primeira opção a identificada como mais útil, pelo autor.

### 3.4.5 Execução

Cohn (2005) refere que nos projetos ágeis não existe uma divisão formal em fases de desenvolvimento, tal como acontece no modelo em cascata seguido pelas metodologias tradicionais. Embora possa existir algum trabalho inicial rápido de desenho ou modelação, desde que a execução é iniciada todo o trabalho (análise, desenho, codificação, testes, etc.) acontecem concorrencialmente em cada iteração.

As iterações (*sprints* em scrum) têm um tempo fixo constante e determinado, e o trabalho de uma iteração termina sempre na data programada, mesmo que todas as funcionalidades não sejam concluídas a tempo.

Cohn (2005) refere ainda que as iterações normalmente são de uma a quatro semanas e que a sua característica principal é a transformação de requisitos em *software* funcional e testado, com as características para ser potencialmente disponibilizado aos utilizadores. (o *software* não é obviamente entregue ao utilizador até estar no ponto de desenvolvimento que o permita, o que se pretende é que cada sprint produza um elemento de *software* o mais fechado e testado possível e que o seu nível de qualidade esteja num patamar elevado).

(Cohn, 2005) refere que como uma iteração não é normalmente suficiente para criar funcionalidades suficientes para satisfazer as necessidades do cliente, estas são agrupadas em *releases* que têm normalmente uma duração de 1 a 6 meses.

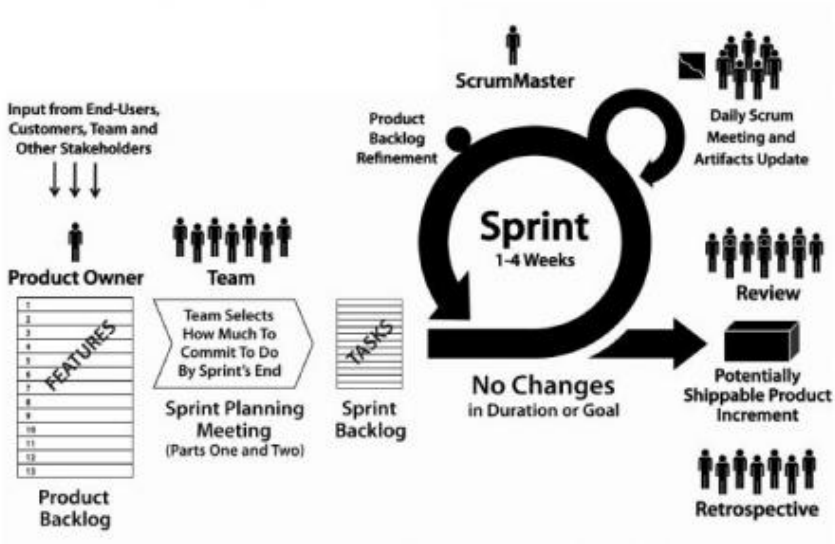



**Figura 22 - Os processos do Scrum. (Larman, 2008)**

### 3.4.6 Monitorização e controlo

#### 3.4.6.1 Velocity

A *velocity* é uma medida que permite obter o ritmo do progresso de uma equipa. Este indicador é calculado pelo somatório do número de *story points* assignados a cada *user story* que a equipa completou durante a iteração (por exemplo, se durante a iteração a equipa completar 3 histórias com 5 pontos cada, a velocidade da equipa é 15). A grande vantagem desta informação é permitir prever o trabalho que será realizado em cada iteração futura. Se uma equipa completou 10 pontos nas últimas iterações, é expectável que consiga concluir o mesmo número de pontos nesta iteração. Como estas estimativas são relativas, esta presunção é tão verdadeira para 2 histórias com 5 pontos cada como para 5 histórias com 2 pontos cada.
A *velocity* é a principal medida do progresso da equipa. (Cohn, 2005)

#### 3.4.6.2 Monitorização ao nível da *Release*

Cohn (2005) refere que o plano de *release* estabelecido no início define o número de iterações a realizar bem como a quantidade de *story points*. Este plano não é fechado e poderá mudar ao longo do tempo, no entanto, é essencial que em qualquer momento se possa monitorizar em que estado está o trabalho em relação ao objetivo. Para obter esta informação é necessário considerar:

- O progresso realizado pela equipa
- Existência de alterações no âmbito do projeto realizadas pelo *product owner*
- Existência de alterações às estimativas realizadas pelos membros da equipa, derivadas da experiencia que foi obtida na iteração anterior

##### *3.4.6.2.1 Release Burndown Charts*

Estes gráficos mostram a quantidade de trabalho por realizar no início de cada iteração. É um indicador visual do progresso da equipa para o atingimento dos objetivos relacionando o número de *story points* por realizar com as iterações. Cohn (2005).

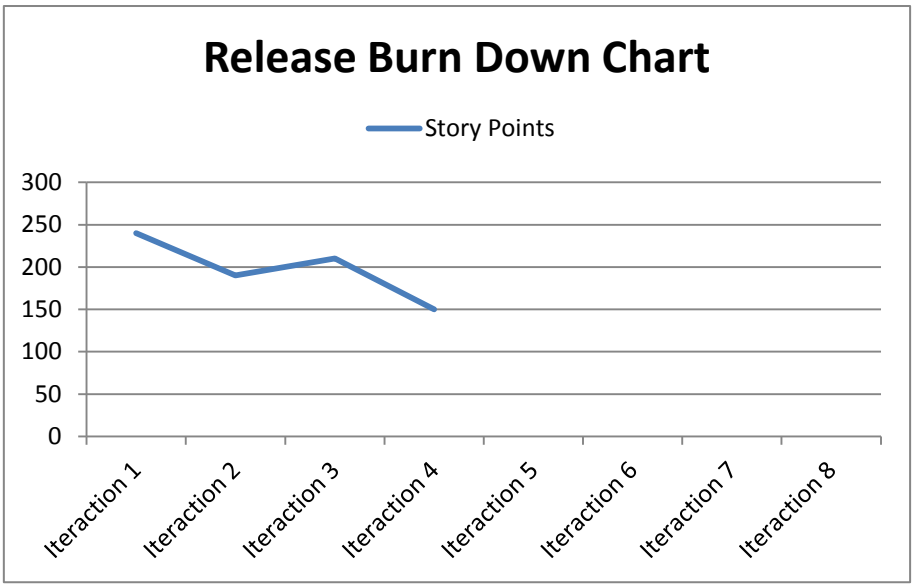



**Figura 23 - *Release* Burn Down Chart. Adaptado de Cohn (2005)**

Neste gráfico, pode ver-se que o trabalho a realizar no fim da segunda iteração é maior que no início. Isto pode significar que as estimativas foram revistas pela equipa, ou que o *product owner* adicionou *user stories* à *release*.

O desenho deste gráfico permite-lhe mostrar as duas variáveis principais para determinar se um projeto esta a ser realizado de acordo com o planeado:

- Quanto trabalho falta realizar
- Ritmo de progresso da equipa independentemente das alterações ao âmbito do projeto

No exemplo apresentado na figura seguinte é possível determinar que o projeto não vai ser concluído nas 8 iterações planeadas (pois houve um aumento nos *story points*) mas também quando se espera que esteja terminado

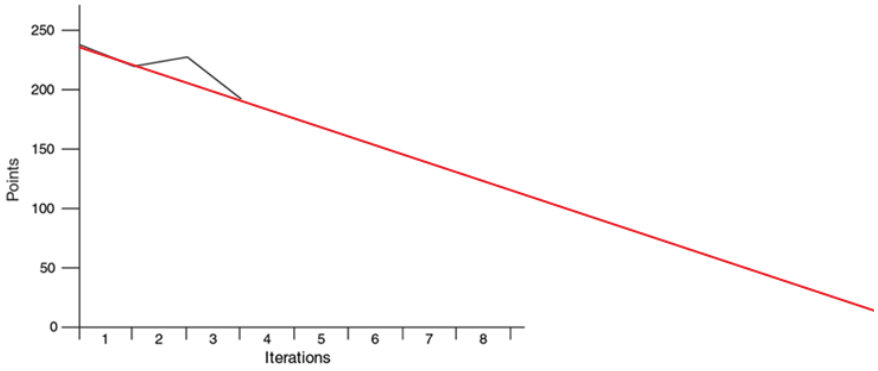



**Figura 24 - Previsão de conclusão no Release Burn Down Chart. Fonte: Cohn (2005)**

### *3.4.6.3 Monitorização ao nível da Iteração*

Segundo Cohn (2005), para além da monitorização realizada ao nível da *release*, as metodologias ágeis permitem um controlo ao nível do trabalho realizado na iteração.

Este controlo é realizado através de duas ferramentas:

- O quadro de tarefas
- *Iteration Burbdown Charts*

#### *3.4.6.3.1 Quadro de tarefas*

Cohn (2005) refere que o quadro de tarefas disponibiliza à equipa um mecanismo conveniente para organização do trabalho, e permite obter uma imagem imediata da quantidade de trabalho que está por realizar.

Cohn (2005) refere ainda que este quadro permite à equipa ter uma grande flexibilidade na forma como organiza o seu trabalho. Nas metodologias ágeis, os recursos não são assignados às tarefas antecipadamente, mas apenas quando estão prontos para que estes lhes possam pegar. Isto significa que no início da iteração é típico que existam várias tarefas sem nenhum elemento associado.

Este quadro pode ser uma ferramenta de *software*, ou um quadro branco ou de cortiça onde são colocados cartões com as "histórias de utilizador" a implementar e as correspondentes tarefas identificadas no planeamento da iteração. Um exemplo deste quadro é apresentado na figura seguinte:

| Story | To Do | Tests Ready | In Process | To Verify | Hours |
|---|---|---|---|---|---|
| As a user, I can... 5 | Code the... 8<br>Code the... 5<br>Test the... 6 | √ | Code the... SC 6<br>Code the... DC 4 | Code the... LC 4 | 33 |
| As a user, I can... 2 | Code the... 8<br>Code the... 5 | | | | 13 |
| As a user, I can... 3 | Code the... 3<br>Code the... 6 | √ | Code the... MC 4 | | 13 |

**Figura 25 - Quadro de tarefas. Fonte: Cohn (2005)**

As colunas deste quadro representam:

1. *Story*: A funcionalidade a ser implementada. Para cada funcionalidade é apresentado o número correspondente aos *story points* necessários para implementação.
2. *To do*: As tarefas necessárias para implementar a funcionalidade. Para cada tarefa é apresentado o número de horas necessário para implementação
3. *Test Ready*: Indica se foram cumpridos os testes de aceitação funcionais por parte do *product owner* e como tal o trabalho de desenvolvimento pode ser iniciado
4. *In Process*: Contém as tarefas que já estão em desenvolvimento, Cada elemento da equipa vai assumindo tarefas disponíveis à medida que vai concluindo as anteriores.
5. *To Verify*: Tarefas que estão concluídas mas que carecem de validação ou testes adicionais
6. *Hours*: Número de horas previstas total para conclusão da funcionalidade. Os valores de horas devem se atualizados pelos membros da equipa caso percebam que as estimativas estavam incorretas, sendo atualizado também o valor total de horas.

A imagem seguinte apresenta alguns exemplos de quadros de tarefas reais, físicos e implementados em *software:*

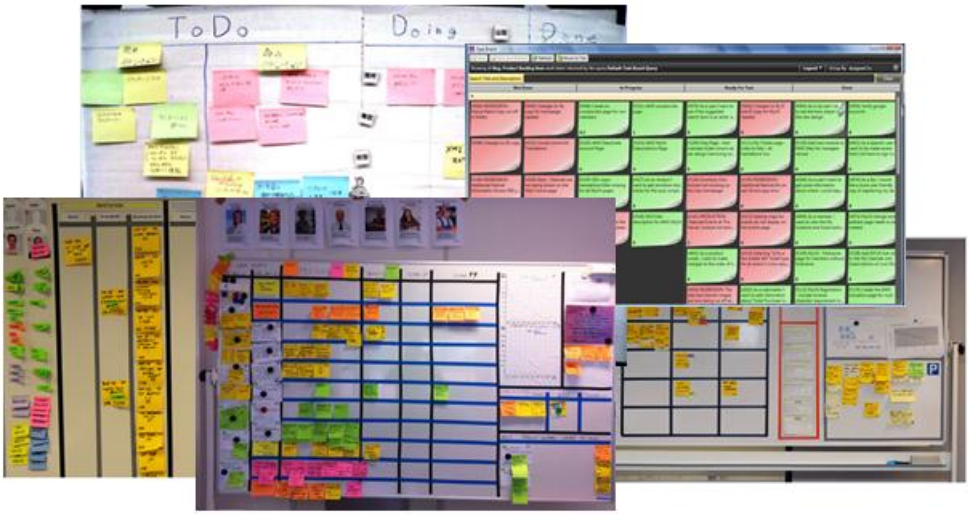



**Figura 26 - Exemplos de quadros de tarefas. Colagem do Autor a partir de fontes diversas disponíveis em images.google.com**

#### *3.4.6.3.2 Iteraction Burndown Charts*

Cohn (2005) refere que este gráfico permite obter uma imagem do estado de uma iteração, embora não faça sentido para iterações inferiores a 2 semanas. Este gráfico mostra as horas de trabalho em falta por dia, tal como representado na figura seguinte:

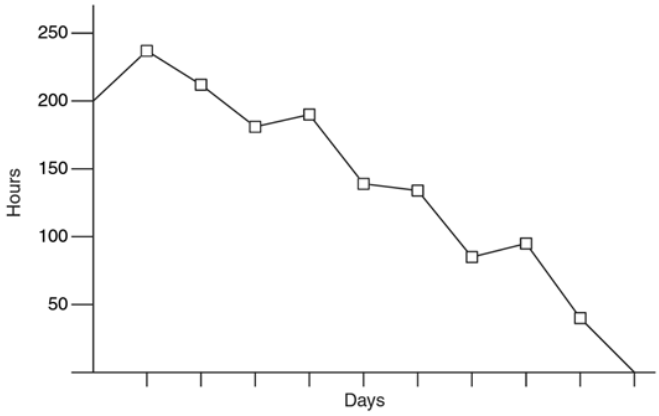



**Figura 26 -Iteraction Burndown Charts. Fonte: Cohn (2005)**

#### 3.4.6.3.3 Velocidade individual

Cohn (2005) adverte que apesar de ser possível medir a velocidade individual de cada membro da equipa, isto não deve ser feito. Toda a metodologia de gestão ágil foi criada para incentivar o espírito de equipa, de colaboração e entreajuda. A criação de um elemento de competição entre os membros da equipa iria comprometer este objetivo.

O autor adverte ainda que as histórias de utilizador devem ser escritas de uma forma que as obrigue a ser trabalhadas por mais de uma pessoa (por exemplo um designer, um programador, um técnico de bases de dados e um responsável por testes)

### 3.4.7 Relatórios e comunicação

Cohn (2005) indica que as metodologias ágeis baseiam-se numa comunicação honesta e aberta. Se um programador percebe que uma tarefa vai demorar mais tempo do que estava previsto, tem de sentir confiança para o poder comunicar ao resto da equipa, incluindo a direção.

Cohn (2005) refere ainda que é também muito importante que a comunicação acerca das estimativas e planeamento seja realizada em dois sentidos, encorajando sempre o diálogo e a

discussão acerca do plano, de forma a tentar garantir sempre que se chega ao melhor plano possível para o conhecimento atual, para acrescentar o máximo valor possível à organização.
O objetivo é iterar e refinar os planos com base nos feedbacks e em novos conhecimentos, promovendo diálogos e não apresentações num só sentido.

### 3.4.7.1 Comunicação do plano

Cohn, (2005) refere que nas metodologias tradicionais, o plano de projeto é comunicado utilizando gráficos *Gantt*. Nos projetos de desenvolvimento de *software* estes gráficos têm ganho alguma má reputação, mas isto deve-se à sua utilização incorreta (do ponto de vista ágil) para prever a calendarização e coordenar tarefas.
Nas metodologias ágeis este tipo de gráficos continua a fazer sentido, mas orientados a funcionalidades e não a tarefas, tal como representado na figura seguinte:

**Figura 27 - Cronograma ágil. Fonte: Cohn, (2005)**

Cohn, (2005) salienta que as metodologias ágeis favorecem as equipas e não os indivíduos, da mesma forma responsabilizam a equipa pela boa execução do plano. Por este motivo as funcionalidades não aparecem associadas a recursos e a sua execução é da responsabilidade de toda a equipa.

#### 3.4.7.1.1 Comunicação do progresso

Cohn (2005) indica que a principal ferramenta para comunicação do progresso são os *release burndown charts*, que representam a relação entre o trabalho que falta realizar e a velocidade da equipa.
Esta informação permite facilmente calcular o número de iterações necessário para conclusão do projeto, bastando dividir os pontos em falta pela velocidade da equipa (o autor salienta que

a velocidade da equipa não é sempre a mesma e que devem ser utilizados cenários com diferente velocidades, com base no historial da equipa).

### 3.4.8 Validação das metodologias ágeis

Chow & Dac-Buu (2008) referem que apesar do *software* ser fundamental para todas as vertentes do mundo moderno, os seus projetos e a forma como são geridos apresentam problemas já identificados e amplamente conhecidos, entre os quais o facto de grande parte resultar em projetos atrasados, falhados ou rejeitados pelos utilizadores e que muitos dos que chegam ao fim com sucesso tendem a necessitar de muito trabalho de manutenção e correção, Os autores referem o surgimento das metodologias ágeis, e o seu objetivo de resolver estes problemas, fazendo a gestão de uma forma inteiramente diferente.

Para determinar se os fatores críticos de sucesso defendidos por estas metodologias tinham efeitos práticos reais, estes autores realizaram um estudo quantitativo sobre 109 projetos geridos por metodologias ágeis, realizados em 25 países diferentes.

Os resultados deste estudo evidenciaram que muitos dos fatores críticos de sucesso defendidos por estas metodologias, e expressados no manifesto ágil, são de facto reais e mensuráveis com técnicas estatisticas através de várias técnicas de regressão.

Os autores concluíram que as metodologias ágeis permitem uma forma de gestão de projetos de desenvolvimento de *software* adequada e que os principais fatores críticos de sucesso são:

- Seguimento de uma estratégia de entregas regulares
- Utilização correta das técnicas de gestão ágil
- Criar e manter uma equipa de qualidade

## 3.5 Os problemas da gestão tradicional do ponto de vista ágil

Cohn (2005) refere que um problema crítico das metodologias tradicionais em relação ao planeamento de projetos de *software*, é que se foca em atividades e não em funcionalidades. Nestas metodologias tradicionais, as atividades são agrupadas em "*Work Breack Down Structures*" ou em gráficos de *Gantt*, e avaliação do sucesso é feito pela determinação do seu atingimento. O problema desta abordagem é que o cliente não tira qualquer partido da

conclusão de uma atividade. Por outro lado, uma funcionalidade é uma unidade de trabalho que tem valor para o cliente.

Cohn (2005) apresenta algumas razões para os atrasos nos projetos de desenvolvimento de *software* baseados em atividades serem tão comuns:

- As atividades não terminam antes do tempo planeado: tal como estipulado pela lei de *Parkinson* (Parkinson, 1955) "O trabalho expande-se de forma a preencher todo o tempo disponível para a sua realização". Isto significa que as pessoas tendem a demorar o tempo que consideram que lhes foi autorizado para a realização de uma atividade. Mesmo que a tarefa seja realizada antes de tempo, o seu executor tende a gastar o tempo remanescente a investigar ou a dedicar-se a outras atividades. Em algumas organizações, o fecho de uma tarefa antes do tempo pode levar o gestor a pensar que o planeamento foi mal feito, ou a esperar que todas as restantes tarefas sejam realizadas antes do tempo.

- O atraso de uma atividade é passado para outras tarefas calendarizadas: As atividades que estão dependentes de uma atrasada irão começar atrasadas, e uma atividade só poderá começar mais cedo se todas as tarefas que a precederem terminarem antecipadamente.

- As atividades não são independentes: As atividades dizem-se independentes se a duração de uma não tem influência sobre a duração da outra. Por exemplo na construção de uma casa, o tempo necessário para criar as fundações é independente do tempo necessário para pintar as paredes. Nas atividades independentes, o atraso numa pode ser compensado com a conclusão antecipada de outra. No entanto a maior parte das atividades relacionadas com o desenvolvimento de aplicações de *software* não são independentes, e se uma demora mais tempo de que o previsto, é muito provável que as outras semelhante também demorem mais tempo do que o previsto.

Cohn (2005) apresenta ainda alguns motivos para a gestão de projetos por metodologias tradicionais falhar:

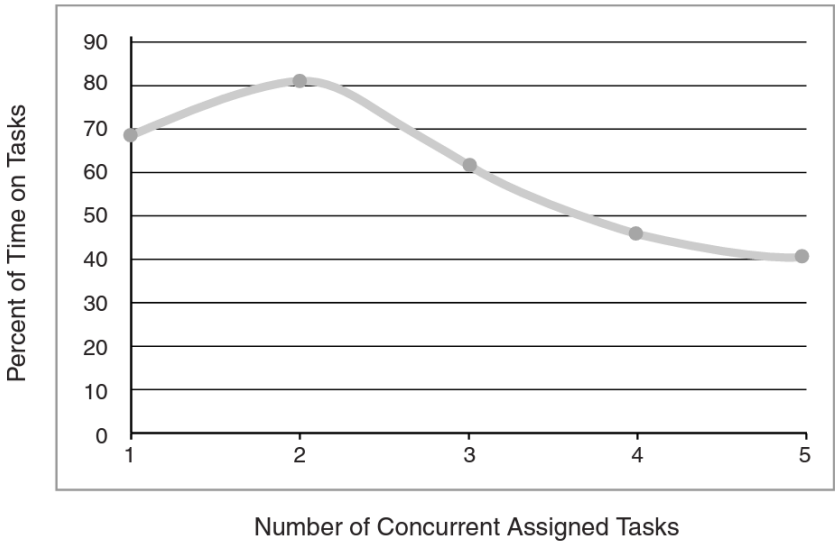
- As atividades não são desenvolvidas por prioridade: No planeamento tradicional, o trabalho definido no plano do projeto não está priorizado pelo valor que tem para os seus utilizadores e clientes. Os planos tradicionais são criados com a assunção de que o cliente só terá acesso à solução quando o projeto for concluído, pelo que a ordem de sequenciação das atividades é arbitrária para o seu cliente. Normalmente quando o projeto se aproxima do fim do tempo planeado e está atrasado, há a tentação de reduzir o âmbito, retirando algumas funcionalidades, entre as quais podem estar as mais importantes para o cliente.

- A gestão tradicional ignora a incerteza: A gestão tradicional tende a ignorar a incerteza e a assumir que os requisitos inicialmente definidos levam a uma satisfação total do cliente, e que este não vai mudar de opinião ou ter novas necessidades durante o período coberto pelo plano. A gestão tradicional também tende a ignorar a incerteza acerca de como será realizado o produto e admite que é possível assignar estimativas precisas para trabalho impreciso. É impossível identificar todas as atividades que vão ser necessárias durante um projeto, no entanto esta realidade é constantemente esquecida nos planos que são realizados.

- As estimativas transformam-se em compromissos: Cada estimativa tem associada uma probabilidade de a execução ser concluída no tempo estimado. A gestão tradicional tem tendência a assumir o tempo associado a uma estimativa como um compromisso, ignorando esta probabilidade, e é impossível associar um compromisso a uma probabilidade

Cohn (2005) refere ainda que outro problema comum das metodologias tradicionais é a alocação de várias tarefas em simultâneo a um mesmo recurso. O autor refere que a realização de múltiplas tarefas em simultâneo tem um efeito desastroso sobre a produtividade, e refere um estudo realizado para avaliar este impacto[3] que concluiu que o tempo que é gasto na realização de trabalho sobre uma tarefa diminui drasticamente quando se está a trabalhar em mais do que duas tarefas em simultâneo. A figura seguinte apresenta o resultado deste estudo:

[3] Clark, Kim B., and Steven C. Wheelwright (1993). Managing New Product and Process Development: Text and Cases. The Free Press.




A atribuição de múltiplas tarefas ao mesmo recurso encoraja um elevado nível de utilização dos indivíduos que compõem a equipa do projeto sem deixar folgas para a variabilidade típica deste tipo de atividades. O autor compara a alocação dos recursos a 100% da sua capacidade com uma autoestrada ocupada em 100% da sua capacidade, em qualquer um dos casos o progresso torna-se impossível.

## 3.6 Conclusões da revisão de literatura

A partir da revisão de literatura foi possível identificar algumas das características específicas dos projetos de *software*, e conhecer alguns dos elementos que contribuem para os conhecidos problemas nos seus resultados

Questionados sobre estas temáticas, um conjunto de gestores de projeto identificaram como os desafios mais frequentes o cumprimento de prazos, o cumprimento dos restantes constrangimentos e a comunicação eficaz entre grupos de trabalho.

A "obsessão" dos gestores de projeto pelo cumprimento destes constrangimentos foi de resto identificada como uma das maiores causas de insucesso na gestão de projetos de *software*.

Esta ideia foi confirmada através de uma investigação que procurou identificar os fatores críticos de sucesso para estes projetos, tendo concluído que para a grande maioria dos que foram concluídos com sucesso, o fator crítico principal, tanto para os membros da equipa, como para os gestores de projeto, foi o "atingimento dos requisitos do utilizador". Esta

investigação permitiu ainda concluir que para a grande maioria dos projetos que falharam, os fatores críticos de sucesso foram o "cumprimento do orçamento" e o "cumprimento da calendarização"

De acordo com a literatura estudada, e atendendo à especificidade dos projetos de desenvolvimento de *software*, a sua probabilidade de sucesso é muito maior se forem considerados critérios de sucesso relacionados com o atingimento de requisitos e satisfação dos clientes.

Este trabalho permitiu também concluir que o ativo mais importante do projeto são as pessoas que fazem parte das equipas de projeto e que a manutenção da sua motivação, do nível de participação que têm, na responsabilidade do projeto e o consequente sentimento de pertença e identificação com os objetivos, são fundamentais para a realização de projetos com sucesso.

A revisão de literatura permitiu também obter uma caracterização aprofundada sobre duas vertentes de metodologias de gestão de projetos: a tradicional cujo corpo de conhecimentos se encontra explanado no PMBOK e as ágeis, com surgimento mais recente, mas surgiram devido aos problemas específicos da gestão de projetos de desenvolvimento de *software*. Estas metodologias ágeis defendem que o nível de sucesso na gestão de projetos de desenvolvimento de *software* pelas metodologias tradicionais não é satisfatório e propõem uma alternativa pensada de raiz para os abordar.

Após extensa revisão bibliográfica, incluindo uma investigação realizada para verificar a legitimidade dos fatores críticos de sucesso defendidos por estas metodologias (baseados no manifesto ágil), as metodologias ágeis foram consideradas por este trabalho como as mais indicadas para resolver as questões identificadas no diagnóstico da ZTelc e para gerir os projetos de desenvolvimento de *software* em geral, em organizações e equipas com características semelhantes.

# 4 Apresentação da empresa e do problema em estudo

## 4.1 Descrição da empresa

Por questões de confidencialidade, a empresa que serviu de base para a realização deste estudo será designada neste trabalho como ZTelc.

A ZTelc é uma empresa que opera no sector das telecomunicações, presente no mercado nacional desde 1997.

Em termos organizativos, esta empresa conta com cerca de 300 colaboradores, distribuídos por 15 direções funcionais.

## 4.2 Descrição da direção de IT

A direção de IT da ZTelc é constituída por cerca de 40 colaboradores, organizados de acordo com a estrutura apresentada no seguinte organigrama:

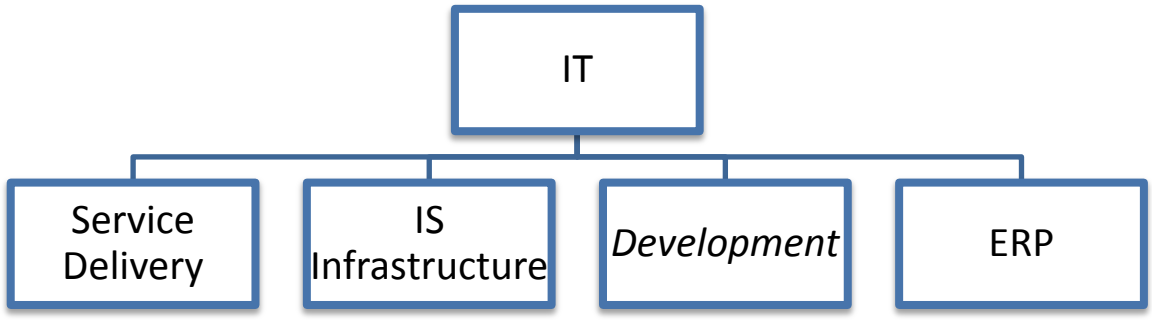



**Figura 28 - Organigrama da direção de IT da ZTelc**

Esta direção é composta por dois subgrupos lógicos, um mais vocacionado para a manutenção das operações diárias que garantem o negócio:

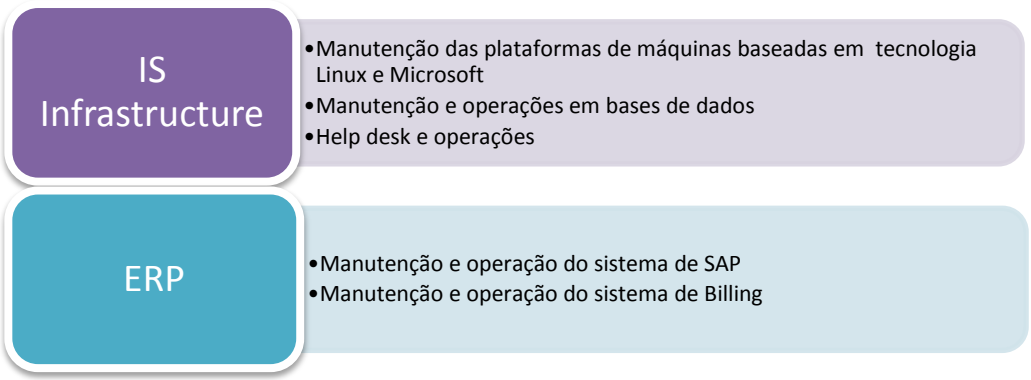



**Figura 29 - Departamentos operacionais da ZTelc**

E o outro mais vocacionado para a criação de valor através da execução de estratégias definidas pela gestão da empresa, e concretizadas sob a forma de projetos:

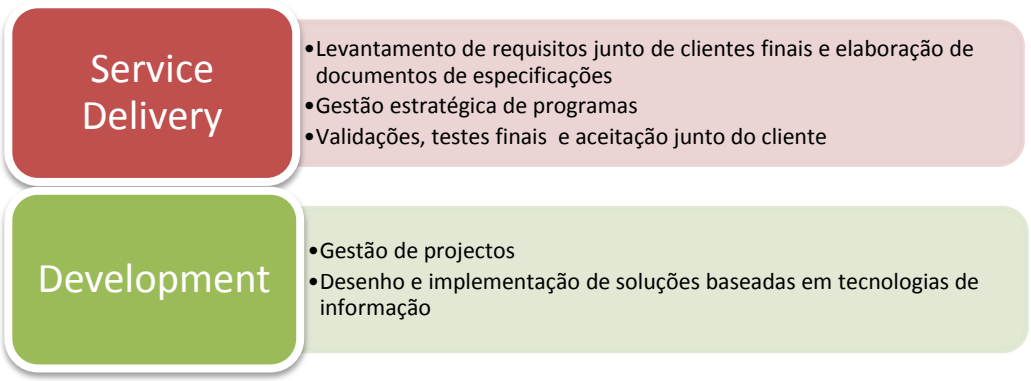



**Figura 30 - Departamentos dedicados a projetos na ZTelc**

## 4.3 Descrição do ambiente de gestão de projetos

As subdireções "*Service Delivery*" e "*Development*" estão alinhadas com a estratégia da organização para a execução de projetos de TI, que pode ser representada pelo seguinte macroprocesso:

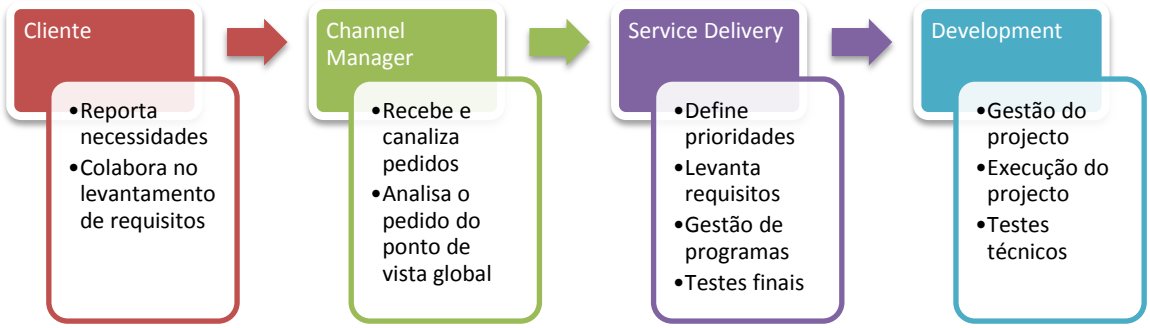



**Figura 31 - Macroprocesso de gestão de projetos na ZTelc**

O cliente é a representação de uma determinada área da empresa que solicita um projeto de desenvolvimento para implementação ou melhoria de um processo organizacional ao seu *Channel Manager*.

Os processos específicos da equipa de desenvolvimento podem ser representados pelo diagrama:

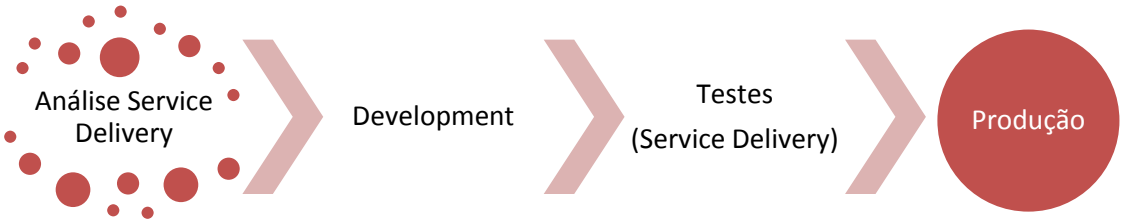



**Figura 32 - Processo de desenvolvimento da ZTelc**

A unidade organizacional "*Development*" corresponde a uma subdireção da direção de IT, e está organizada de uma forma atípica em relação a todas as outras unidades da empresa. Uma vez que foi especificamente pensada para a concretização de projetos:

| Sub Director Development | |
|---|---|
| Chefes de Projecto | Membros de Projecto |
| Chefe 1 | Membro 1 |
| Chefe 2 | Membro 2 |
| Chefe 3 | Membro 3 |
| Chefe 4 | Membro 4 |

**Figura 33 - Estrutura idealizada da equipa de desenvolvimento**

Esta subdireção é constituída por 10 elementos e é uma macro equipa, uma vez que os recursos dependem de um único superior hierárquico, o subdiretor.

Para além desta organização flexível, a subdireção tem uma segunda organização mais tradicional, organizada por especialidades técnicas e que corresponde a uma organização formal precedente. No entanto, esta é a organização em uso, pois os membros da equipa continuam a trabalhar exclusivamente em projetos relacionados com a sua área funcional, e não existe o perfil de Gestor de Projeto:

| Sub Director Development | | |
|---|---|---|
| Plataforma 1 | Plataforma 2 | Plataforma 3 |
| Chefe Equipa | Chefe Equipa | Chefe Equipa |
| Membro 1 | Membro 1 | Membro 1 |
| Membro 2 | Membro 2 | Membro 2 |
| Membro 3 | Membro 3 | Membro 3 |

**Figura 34 - Estrutura real da equipa de desenvolvimento**

### 4.3.1 Planeamento de Projetos

O "*Channel Manager*" é responsável por avaliar a relevância do pedido e a sua pertinência face a todos os outros pedidos realizados pelas diferentes áreas da organização. Caso este considere que o projeto é relevante, é feito um pedido ao "*Service Delivery*" para a sua formalização.

O "*Service Delivery*" recebe os pedidos dos vários "*Chanel Managers*" da organização, e determina a sua prioridade através de um conjunto de parâmetros pré-estabelecidos pela direção.

Com base nestes elementos é atribuído um "*Scoring*" ao projeto, que irá indicar a sua importância e prioridade de desenvolvimento. O "*Service Delivery*" fica então responsável por comunicar com o cliente e depurar as necessidades e objetivos concretos do projeto, elaborando um documento denominado "Manual de Projeto" onde é colocado o pedido do cliente bem como todo o trabalho de análise realizado pela equipa do "*service delivery*", e que irá corresponder ao conjunto de requisitos a implementar. Uma vez concluído o levantamento de requisitos, o Manual de Projeto é disponibilizado a um responsável técnico (um elemento da equipa de desenvolvimento) de cada subequipa envolvida no projeto, para que realize a análise técnica. Esta análise corresponde a uma especificação detalhada da solução final que

será implementada e à tradução dos requisitos em tarefas específicas, que correspondem ao 3º nível da WBS, e à estimação do seu esforço.

Com base nesta informação o "*Service Delivery*" valida com o cliente que a solução final é o desenvolvimento que se pretende. Caso não seja é repetido todo o processo, caso contrário elabora um cronograma com base nas tarefas e estimativas elaboradas pelos elementos técnicos.

Após concluído este processo, o projeto é colocado num estado que o identifica como estando pronto para o início do desenvolvimento.

### 4.3.2 Execução, controlo e monitorização

O subdiretor da direção, em conjunto com os "chefes de equipa" realiza semanalmente uma reunião onde é verificado o estado de cada projeto em curso, sendo este reportado verbalmente. Nesta reunião são também decididos os próximos projetos a implementar, a partir da lista de projetos cuja análise está concluída pelo "*Service Delivery*" e pelo "*scoring*" atribuído na fase de planeamento.

Durante o desenvolvimento não existe um processo formal de comunicação ou avaliação de estado do projeto, sendo feito ocasionalmente um ponto de situação verbal.

### 4.3.3 Relatórios e comunicação

Não existe a figura do gestor de projeto, sendo cada projeto coordenado pelo responsável funcional de cada subequipa, pelo *service delivery* e pelo subdiretor da área. Não existem processos formais de passagem de informação dos membros das equipas para a gestão de projeto. A comunicação para a direção da organização é baseada nos cronogramas.

# 5 Diagnóstico da situação atual

## 5.1 Resultados das entrevistas

Através de entrevistas semiestruturadas realizadas aos diversos elementos da equipa (Anexo A) foi possível identificar as suas principais dificuldades e necessidades.

As entrevistas foram conduzidas de forma a obter informação agrupada em dois grupos lógicos: (1) Dificuldades e (2) Equipa.

Da análise de todas as entrevistas, foi feita a consolidação das respostas numa matriz (Anexo C) e sintetizadas as questões mais relevantes a endereçar neste estudo nos seguintes quadros de resumo:

| Dificuldades identificadas durante a fase de planeamento |
|---|
| 1 - Requisitos insuficientes ou não detalhados o suficiente para cumprir os reais objetivos do cliente ou as necessidades de informação da equipa |
| 2 - Não envolvimento da equipa técnica na realização dos requisitos e planeamento, resultando em planos incompletos ou incongruentes |
| 3 - Tempo para a realização cuidada do planeamento técnico |

**Tabela 10 - Dificuldades identificadas na fase de planeamento**

| Dificuldades identificadas durante as fases de execução, monitorização e controlo |
|---|
| 1 - Interrupção constante da equipa de desenvolvimento para pegar noutros projetos que se tornaram mais prioritários ou dar suporte a projetos já em produção |
| 2 - Alterações ou dúvidas em relação aos requisitos durante o desenvolvimento |
| 3 - Afetação de elementos da equipa a mais do que um projeto em simultâneo |
| 4 - Isolamento das subequipas que trabalham de forma isolada e com diferentes prioridades |

**Tabela 11 - Dificuldades identificadas na fase de execução, monitorização e controlo**

As interrupções permanentes durante o desenvolvimento de um projeto, quer por alterações de requisitos, quer para reavaliação de negócio ou para transferência de esforço para outro projeto devido a alterações estratégicas de prioridades, constituem a maior dificuldade reportada, tendo sido identificada por todos os elementos entrevistados.

| Dificuldades identificadas ao nível dos recursos humanos |
|---|
| 1 – Acomodação dos membros da equipa por especialização tecnológica, formando subequipas que impedem o verdadeiro trabalho em equipa |
| 2 - Falta de motivação, falta de sentimento de pertença aos projetos |
| 3 - Inexistência de um Gestor de Projetos formal e com responsabilidade sobre toda a equipa |

**Tabela 12 - Dificuldades identificadas ao nível dos Recursos Humanos**

| Necessidades de informação e comunicação |
|---|
| 1 - Informação geral sobre os projetos da equipa |
| 2 - Feedback do cliente sobre o que está a ser desenvolvido |
| 3 - Conhecimento do trabalho realizado pelos outros elementos da equipa e incentivo à entreajuda entre todos os membros |

**Tabela 13 - Necessidades de informação e comunicação identificadas**

| Pontos fortes da equipa |
|---|
| 1 - Boa capacidade de adaptação |
| 2 - Conhecimentos técnicos abrangentes |
| 3 - Boa capacidade de autoformação |

**Tabela 14 - Pontos fortes da equipa identificados**

| Pontos fracos da equipa |
|---|
| 1 - Equipa "partida" em subequipas isoladas em termos funcionais e de informação |
| 2 - Grande aversão ao risco |
| 3 - Falta de metodologias e ferramentas para uma gestão de projetos eficiente |

**Tabela 15 - Pontos fracos da equipa identificados**

| Sugestões de melhoria |
|---|
| 1 - Criação efetiva do perfil de Gestor de Projetos |
| 2 - Redesenho da estrutura da equipa para um modelo mais funcional e comunicante |
| 3 - Escolha de uma metodologia de gestão de projetos, formação dos membros da equipa e cumprimento dos seus preceitos de forma rigorosa |
| 4 - Criar mecanismos de partilha de informação e colaboração entre todos os elementos da equipa |

**Tabela 16 - Sugestões de melhoria**

## 5.2 Análise quantitativa

Através da análise de documentação interna que foi disponibilizada para a realização deste trabalho foi possível obter alguma informação quantitativa acerca dos projetos realizados no último ano.

### 5.2.1 Projetos realizados

**Figura 35 - Projectos concluídos / Cancelados**

- Foram concluídos 77 projetos
- Foram cancelados 7 projetos para os quais já tinham sido realizados 19 dias de trabalho

Em termos médios, os projetos concluídos apresentaram as seguintes durações por fase:

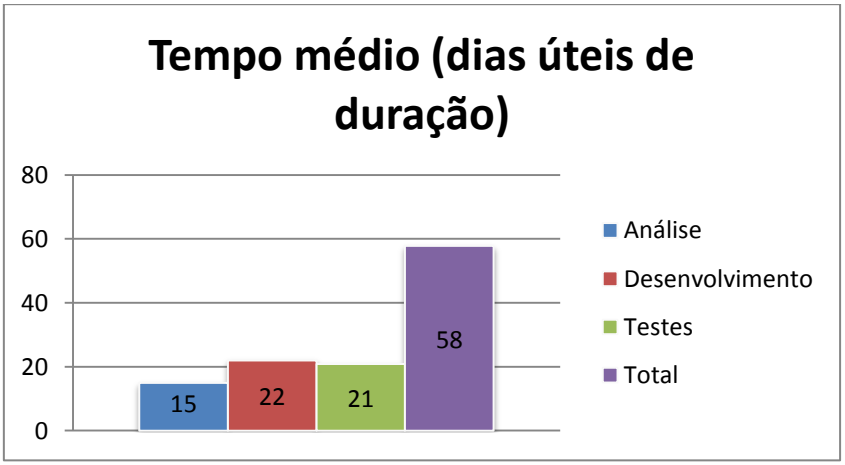



**Figura 36 - Tempo médio (dias úteis)**

Os projetos realizados por esta equipa têm uma duração média total de 58 dias úteis.

### 5.2.2 Composição das equipas por projeto

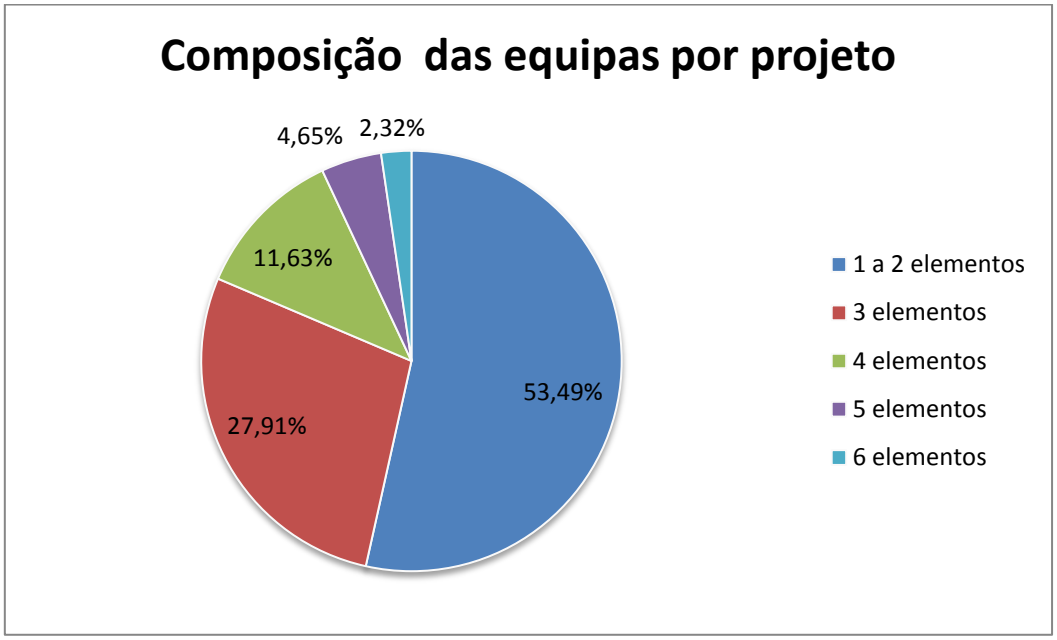



**Figura 37 - Composição das equipas por projeto**

Verifica-se que a grande maioria dos projetos realizados é composto por equipas muito pequenas. Através de entrevistas complementares, foi possível determinar que esta situação ocorre devido a:

- Projetos orientados às plataformas tecnológicas de base, as quais são trabalhadas apenas por um pequeno número de elementos.
- Os projetos contêm um número pequeno de funcionalidades que na maior parte das vezes não cruzam as diversas plataformas tecnológicas de base.

### 5.2.3 Tempo de Análise

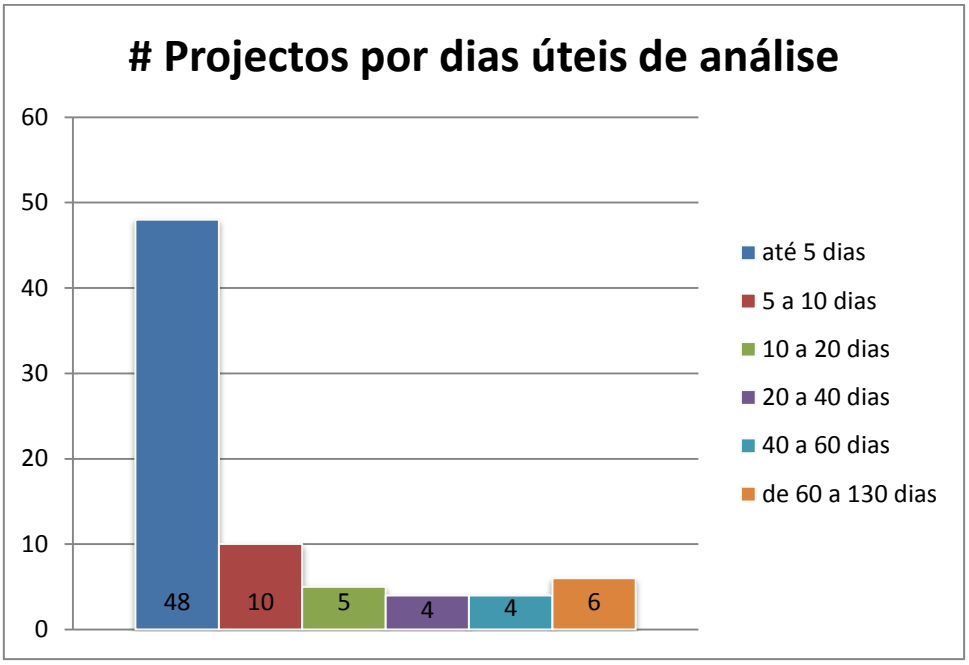



**Figura 38 - Número de projetos por dias úteis de análise**

Através deste gráfico é possível verificar que a análise da maioria dos projetos é realizada num período que não excede os 5 dias. No entanto, existem alguns projetos cuja análise é mais longa, passando mesmo os 100 dias. Através de entrevistas aos elementos responsáveis por esta fase foi possível determinar as causas destas situações:

- Em muitos casos o cliente não sabe exatamente que solução pretende ao colocar um pedido, existindo um longo fluxo de comunicação até se chegar a um conjunto de requisitos
- A equipa não tem disponibilidade para realização dos planeamentos técnicos, ficando o projeto a aguardar esta informação durante algum tempo.
- O projeto fica em "fila de espera" pois os recursos do "*Service Delivery*" estão a realizar a análise de outros projetos.

### 5.2.4 Tempo de desenvolvimento

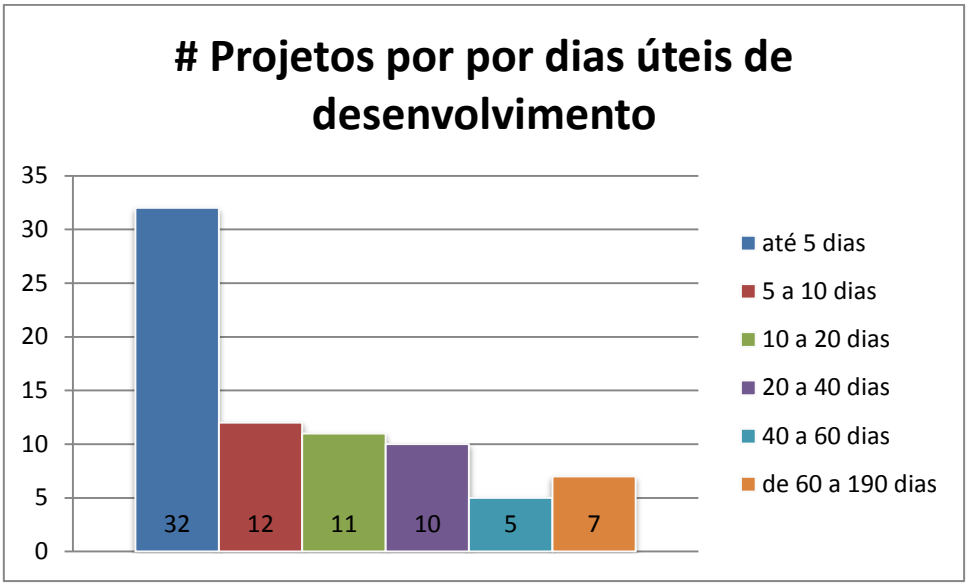



**Figura 39 - Número de projetos por dias úteis de desenvolvimento**

Um elevado número de projetos tem um tempo de execução muito curto (inferior a 5 dias). Esta informação complementada pelo número de projetos realizados por equipas de 1 a 2 elementos indica que estão a ser criados projetos com âmbito excessivamente pequeno e pouco transversal. Esta informação é coerente com os resultados das entrevistas.

Verifica-se também a existência de múltiplos projetos com tempos de desenvolvimento muito superiores ao valor médio (22 dias). Através de entrevistas à equipa de projeto foram identificadas as seguintes causas para esta situação:

- Alteração de requisitos durante a realização do projeto, obrigando a novos períodos de analise durante a execução
- Paragem do projeto por mudança de focos para outro projeto que entretanto se tornou mais prioritário
- Dependência de recursos externos à organização

### 5.2.5 Tempo de testes

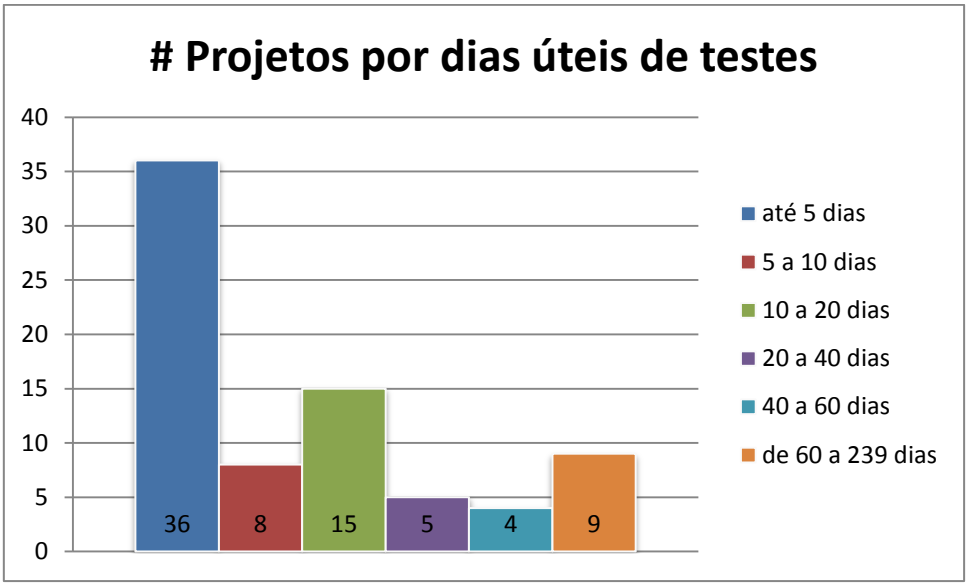



**Figura 40 - Número de projetos por dias úteis de testes**

Seguindo a tendência demonstrada nos gráficos anteriores, também na fase de testes existem múltiplos projetos com tempos muito superiores ao valor médio e ao expectável para um período de testes finais de aceitação. As razões identificadas para esta situação são:

- Os testes dependem do cliente final que nem sempre os realiza atempadamente
- Os testes que resultam em erro ficam pendentes de correção por parte da equipa
- Alguns testes dependem de recursos externos à organização

### 5.2.6 Tempos globais

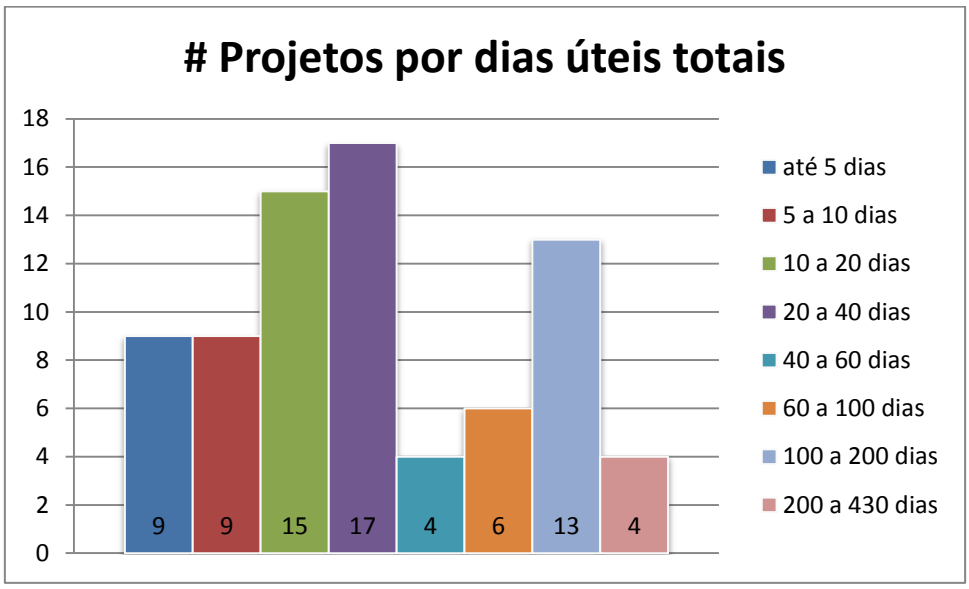



**Figura 41 - Número de projetos por dias úteis totais**

Este gráfico mostra o efeito global de todas as situações identificadas nos gráficos anteriores. Apesar de a equipa ter um tempo médio de realização de projetos de 58 dias, e devido às situações já identificadas, tem projetos com uma duração de até 430 dias.

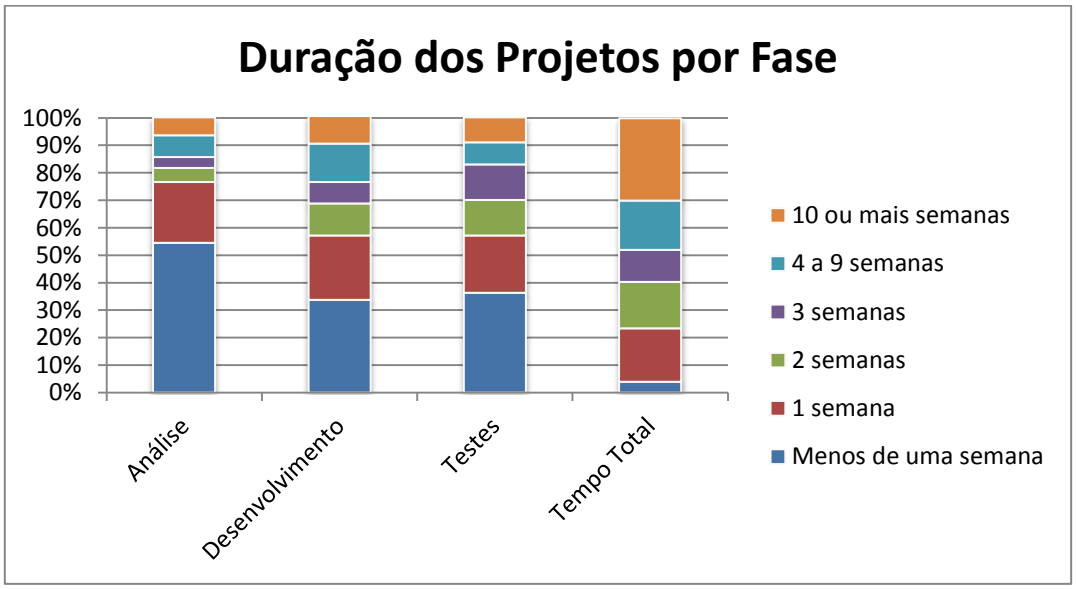



**Figura 42 - Duração dos projetos por fase**

Este gráfico permite uma representação da informação analisada anteriormente, mas com uma representação global e percentual.

### 5.2.7 Variação do tempo de desenvolvimento face ao estimado

Apenas foi possível recolher informação referente à estimativa realizada para 26 projetos. Para estes foi feita uma comparação entre o tempo estimado (apenas de desenvolvimento) e o tempo real:

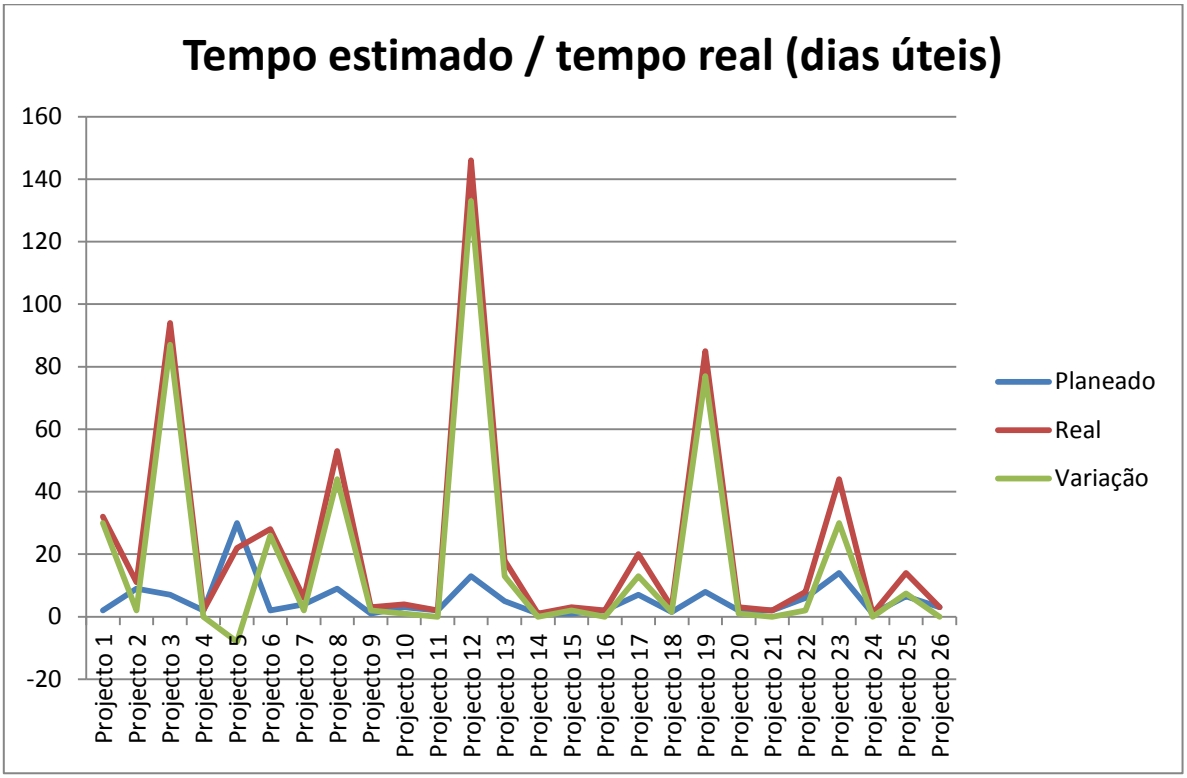



**Figura 43 - Relação entre o tempo estimado e o tempo real (dias úteis)**

Este gráfico permite verificar que dos 26 projetos analisados só um apresenta um tempo de desenvolvimento inferior ao planeado (uma antecipação de 8 dias face a um estimado de 30). A variação entre o estimado e o real apresenta um valor médio de 19 dias úteis, apresentando no entanto vários picos que atingem o valor máximo de 133 dias úteis.

Esta representação confirma e reforça a identificada para a fase de desenvolvimento, sendo as razões justificativas as mesmas. Poderá também indicar que as estimativas estão a ser realizadas de forma irrealista, e com tempos inferiores ao real.

# 6 Propostas de melhorias metodológicas para resolução dos problemas identificados

## 6.1 Recapitulação do problema e objetivos a atingir

No ponto 5 deste trabalho foi realizado um diagnóstico da situação atual da gestão de projetos na ZTelc. Este diagnóstico identificou algumas situações a melhorar, sendo estas sintetizadas no seguinte quadro de objetivos gerais.

| # | Objetivo | Origem | Fases |
|---|---|---|---|
| 1 | Definição clara dos critérios de sucesso para os projetos realizados e reformulação da forma como a direção e a organização consideram a realização de projetos de desenvolvimento de *software* | Literatura<br>Entrevistas<br>Quantitativa | Todas |
| 2 | Agilização da equipa, incentivando o trabalho colaborativo e polivalente com objetivos comuns e promovendo o espirito de equipa e entreajuda a todos os elementos e atribuição formal do perfil de gestor de projetos para apoio e gestão global dos projetos e da informação | Entrevistas | RH<br>Planeamento<br>Execução<br>Controlo<br>Comunicação |
| 3 | Dotar os projetos de requisitos detalhados e suficientes para a realização do projeto | Entrevistas<br>Quantitativa | Planeamento<br>Execução |
| 4 | Rentabilização do uso dos recursos através de integração de requisitos mais abrangentes e horizontais num mesmo projeto, reduzindo o *overhead* de planeamento de testes | Entrevistas<br>Quantitativa | Planeamento<br>Execução |
| 5 | Agilizar o processo de negociação de requisitos com o cliente final, ajudando-o a chegar a uma solução que melhor serve as suas necessidades de negócio | Entrevistas<br>Quantitativa | Planeamento |

| # | Objetivo | Origem | Fases |
|---|---|---|---|
| 6 | Maior envolvimento da equipa técnica na realização dos requisitos e planeamento para assegurar a exequibilidade técnica e as interdependências técnicas e funcionais | Entrevistas | Planeamento<br>Execução |
| 7 | Melhorar as estimativas de tempos para a realização do projeto e disponibilização de tempo para que este trabalho possa ser realizado de forma eficiente | Entrevistas<br>Quantitativa | Planeamento<br>Controlo<br>Monitorização |
| 8 | Criar mecanismos que evitem a interrupção do trabalho da equipa, quer seja por alteração de requisitos, quer seja por alterações de prioridade ou por pedidos de suporte a projetos já fechados, ou que formalizem estas interrupções de forma a não ter impacto negativo no trabalho de desenvolvimento | Entrevistas<br>Quantitativa | Execução<br>Planeamento |
| 9 | Agilizar e facilitar o esclarecimento de dúvidas em relação ao planeamento durante o desenvolvimento do projeto | Entrevistas<br>Quantitativa | Execução |
| 10 | Criar mecanismos para que a equipa não tenha de realizar trabalho em múltiplos projetos em simultâneo | Entrevistas<br>Quantitativa | Execução<br>Planeamento |
| 11 | Melhorar a motivação e sentimento de pertença dos membros da equipa aos projetos através de maior envolvimento no seu planeamento, compreensão dos seus objetivos de negócio e atribuição de responsabilidades dentro da equipa | Entrevistas | RH<br>Planeamento<br>Execução |
| 12 | Criar mecanismos de partilha de informação geral e regular sobre o estado dos projetos e da equipa | Entrevistas | Comunicação |

| # | Objetivo | Origem | Fases |
|---|---|---|---|
| 13 | Criar mecanismos para receção regular de feedback por parte dos utilizadores finais acerca da solução que está a ser implementada | Entrevistas | Comunicação<br>Execução |
| 14 | Agilização dos procedimentos de testes de aceitação por parte do cliente final e de resolução atempadas de problemas identificados nesta fase | Quantitativa | Execução<br>Planeamento |
| 15 | Definição clara de uma metodologia de gestão de projetos, procedendo a formação e comunicação interna acerca das suas regras e funcionamento e observância dos seus preceitos de forma rigorosa | Entrevistas<br>Quantitativa | Todas |

**Tabela 17 - Objectivos de melhoria a atingir**

## 6.2 Propostas de melhoria

### 6.2.1 Proposta de metodologia

O diagnóstico realizado permitiu verificar que a equipa analisada não segue formalmente nenhuma metodologia de gestão de projetos, embora siga um modelo mais tradicional. O ciclo de vida corresponde ao modelo em cascata, sendo a fase de análise e planeamento a que está mais elaborada, existindo uma política de priorização e análise de requisitos, os quais são trabalhados e constituem um plano de projeto formal.

No entanto foram identificados problemas em todas as fases, sendo a grande maioria deles típicos de equipas de desenvolvimento de *software*.

De acordo com a investigação bibliográfica que foi realizada, as metodologias de gestão ágil identificam e propõem soluções específicas para resolver todas estas situações.

Neste caso concreto, e uma vez que não existe uma metodologia transversal e oficial em uso, o risco de passagem para uma metodologia ágil é reduzido.

Por outro lado, os resultados das entrevistas mostram que os desejos manifestados pela equipa se enquadram nos valores das metodologias ágeis.

A proposta deste trabalho é que a organização adira a uma metodologia ágil, nomeadamente o *Scrum*, mas que o faça de uma forma convicta, formando os elementos da equipa bem como os seus clientes internos, e siga os processos metodológicos de forma rigorosa e inflexível.
A flexibilidade desejada deve ser sempre obtida dentro da própria metodologia e não pelo contornar dos seus preceitos.

Um dos fatores chave para o sucesso desta equipa é a compreensão ao nível interno e ao nível da gestão que a definição de sucesso dos projetos é a criação das funcionalidades pedidas pelos utilizadores que são disponibilizadas em cada iteração de acordo com as prioridades definidas pelo *product owner*, e agrupadas por *release*, e não qualquer outro constrangimento externo à metodologia.

Embora a implementação da metodologia não faça parte do âmbito desde trabalho, esta deverá ser realizada cuidadosamente e com o patrocínio da gestão, para que seja aceite e respeitada por todos os membros da equipa e clientes.

### 6.2.2 Especificidades ao nível da equipa

Para que esta equipa possa trabalhar numa metodologia ágil, a sua estrutura terá de ser revista. De acordo com os processos e valores ágeis apresentados neste trabalho, a equipa deverá passar da estrutura real atual:

| Sub Director Development | | |
|---|---|---|
| Plataforma 1 | Plataforma 2 | Plataforma 3 |
| Chefe Equipa | Chefe Equipa | Chefe Equipa |
| Membro 1 | Membro 1 | Membro 1 |
| Membro 2 | Membro 2 | Membro 2 |
| Membro 3 | Membro 3 | Membro 3 |

**Figura 44 - Equipa atual (não ágil)**

Para uma estrutura que privilegie o trabalho em equipa, a colaboração e os processos ágeis:

| Sub Director Development | |
|---|---|
| Gestão de Projectos | Equipa |
| Product Owner | Membro 1 |
| Scrum Master / | Membro 2 |
| | Membro 3 |
| | Membro 4 |
| | Membro 5 |
| | Membro 6 |
| | Membro 7 |
| | Membro 8 |

**Figura 45 - Equipa proposta (ágil)**

Esta proposta considera que:

- Passa a existir uma única equipa sem quebras por especialidades técnicas
- São removidas todas as relações hierárquicas passando a existir apenas "a equipa"
- O perfil de *product owner* poderá ser assumido pelos atuais membros do "*Service Delivery*"
- É atribuído o perfil de *scrum* master a um dos membros da equipa que acumula estas funções com as atuais funções técnicas
- Deverá ser dada a oportunidade e formação adequadas para que todos os membros conheçam todas as plataformas tecnologias. Este processo irá necessariamente ser longo, mas deve ser promovido e explorado. Só assim se poderá tirar verdadeiro partido desta equipa.

### 6.2.3 Especificidades ao nível do planeamento

Atualmente a equipa realiza projetos orientados por tecnologia. O diagnóstico realizado permitiu verificar que muitos desses projetos tinham um âmbito reduzido (poucos dias de desenvolvimento) e que cobriam apenas uma das plataformas tecnológicas, não promovendo a cooperação e o trabalho em equipa.

Com a implementação de um modelo de gestão ágil, o conceito de projeto passa a ser o conceito de *release*, que é o agregador de funcionalidades a implementar.

Para tirar todo o partido da metodologia, as releases devem ser desenhadas para corresponder pelo menos a um mês de trabalho, e devem conter funcionalidades que cruzem todas as plataformas tecnológicas.

### 6.2.4 Especificidades ao nível da execução

A interrupção do trabalho em curso para realização de outras tarefas foi o problema mais reportado nas entrevistas realizadas. Foi também confirmado pelos dados quantitativos que esta situação foi responsável por atrasos em vários projetos.

Com a aplicação de uma metodologia ágil, a interrupção do trabalho a meio de uma iteração deixa de ser possível. Qualquer alteração de requisitos ou de prioridade será atendida na próxima iteração.

O fator chave para o sucesso da implementação desta metodologia, é que os seus processos sejam entendidos e respeitados pela organização. Ao nível da execução é fundamental que as regras sejam seguidas e que todo o trabalho a realizar seja recebido e priorizado pelo product owner, que o colocará numa *release* e numa iteração. Se os clientes entenderam a vantagem do trabalho desta forma e de como as iterações funcionam, irão estar preparados para que qualquer situação prioritária tenha de esperar pelo menos pela próxima iteração. Uma iteração só deve ser interrompida numa emergência.

Para além das interrupções por alteração de requisitos ou de prioridades, também os pedidos de correção de bugs e suporte devem ser tratados da mesma forma, sendo recebidos e priorizados do ponto de vista do negócio em relação a todas as funcionalidades em espera pelo product owner, e passados à equipa através de uma iteração.

### 6.2.5 Propostas de soluções para os objetivos identificados

| # | Descrição | Solução proposta |
|---|---|---|
| 1 | Definição clara dos critérios de sucesso para os projetos realizados e reformulação da forma como a direção e a organização consideram a realização de projetos de desenvolvimento de *software* | Implementação de uma metodologia ágil, nomeadamente o *Scrum*, e respeito rigoroso por todas as suas regras e processos (ver ponto 6.2.1) |
| 2 | Agilização da equipa, incentivando o trabalho colaborativo e polivalente com objetivos comuns e promovendo o espirito de equipa e entreajuda a todos os elementos e atribuição formal do perfil de gestor de projetos para apoio e gestão global dos projetos e da informação | Reorganização da equipa de forma a assumir uma estrutura compatível com os processos e valores das metodologias ágeis (ver ponto 6.2.2) |
| 3 | Dotar os projetos de requisitos detalhados e suficientes para a realização do projeto | Descrição dos requisitos através de User Stories, e permitir que as iterações os detalhem permitindo o inicio imediato do trabalho de desenvolvimento, em vez de aguardar por uma versão final, global e fechada dos requisitos |
| 4 | Rentabilização do uso dos recursos através de integração de requisitos mais abrangentes e horizontais num mesmo projeto, reduzindo o *overhead* de planeamento de testes | Para rentabilizar ao máximo a equipa ágil, cada iteração deve incluir funcionalidade que envolvam os diversos elementos e plataformas |
| 5 | Agilizar o processo de negociação de requisitos com o cliente final, ajudando-o a chegar a uma solução que melhor serve as suas necessidades de negócio | Um dos valores principais dos modelos de gestão ágil é a promoção da colaboração com os clientes e não a negociação contratual. No modelo ágil o cliente não tem que saber exatamente o que pretende, e com base na ideia inicial, a equipa irá |

| # | Descrição | Solução proposta |
|---|---|---|
| | | realizar iterações até chegar à solução para o cliente |
| **6** | Maior envolvimento da equipa técnica na realização dos requisitos e planeamento para assegurar a exequibilidade técnica e as interdependências técnicas e funcionais | No modelo ágil a comunicação e a colaboração são estruturantes. Embora o planeamento seja da responsabilidade exclusiva do product owner, este irá envolver e ouvir a equipa sempre que necessário e pelo menos a cada iteração |
| **7** | Melhorar as estimativas de tempos para a realização do projeto e disponibilização de tempo para que este trabalho possa ser realizado de forma eficiente | Nas metodologias ágeis o planeamento tem lugar e técnicas próprias. |
| **8** | Criar mecanismos que evitem a interrupção do trabalho da equipa, quer seja por alteração de requisitos, quer seja por alterações de prioridade ou por pedidos de suporte a projetos já fechados, ou que formalizem estas interrupções de forma a não ter impacto negativo no trabalho de desenvolvimento | Nas metodologias ágeis há duas regras fundamentais:<br>- O trabalho é realizado em iterações (sprints) que não podem ser interrompidos. Qualquer alteração poderá surgir apenas no próximo ciclo.<br>- A equipa recebe em exclusividade pedidos de trabalho do product owner e no âmbito de uma *release*.<br><br>Se as duas regras forem absorvidas pela organização e respeitadas rigidamente o trabalho nunca será interrompido. |
| **9** | Agilizar e facilitar o esclarecimento de dúvidas em relação ao planeamento durante o desenvolvimento do projeto | Os modelos ágeis promovem a comunicação e o maior envolvimento possível do cliente. Os |

| # | Descrição | Solução proposta |
|---|---|---|
| | | membros da equipa podem falar diretamente com o cliente final e se o cliente estiver informado acerca do funcionamento da metodologia ágil, irá ter todo o interesse em colaborar para poder ter a acesso à sua funcionalidade no final da iteração. |
| **10** | Criar mecanismos para que a equipa não tenha de realizar trabalho em múltiplos projetos em simultâneo | Nas metodologias ágeis o trabalho é realizado por funcionalidades priorizadas, e cada membro é responsável pelas tarefas que pega e realiza. Se se respeitar esta regra e os preceitos definidos no ponto 7, esta situação não pode surgir |
| **11** | Melhorar a motivação e sentimento de pertença dos membros da equipa aos projetos através de maior envolvimento no seu planeamento, compreensão dos seus objetivos de negócio e atribuição de responsabilidades dentro da equipa | Este objetivo corresponde exatamente aos princípios ágeis. |
| **12** | Criar mecanismos de partilha de informação geral e regular sobre o estado dos projetos e da equipa | As metodologias ágeis promovem a comunicação e reuniões regulares, a cada *release*, a cada iteração e a cada dia |
| **13** | Criar mecanismos para receção regular de feedback por parte dos utilizadores finais acerca da solução que está a ser implementada | Este objetivo corresponde a um dos princípios basilares das metodologias ágeis. O feedback é dado no final de cada iteração e os canais de comunicação estão sempre abertos. |
| **14** | Agilização dos procedimentos de testes de aceitação por parte do cliente final e de | Nos modelos ágeis os testes e a respetiva aprovação são realizados a |

| # | Descrição | Solução proposta |
|---|---|---|
| | resolução atempadas de problemas identificados nesta fase | cada iteração. Não é necessário alocar muito tempo para testes de validação finais, pois o projeto foi testado e aprovado a cada incremento que foi realizado. Também não existe o risco que não cumprir os objetivos do cliente pois este acompanhou todo o processo e realizou todos os pedidos de alteração que necessitou |
| 15 | Definição clara de uma metodologia de gestão de projetos, procedendo a formação e comunicação interna acerca das suas regras e funcionamento e observância dos seus preceitos de forma rigorosa | A metodologia proposta é o *Scrum* |

**Tabela 18 - Proposta de soluções para os objetivos identificados**

# 7 Conclusões

A segunda parte deste trabalho focou-se nos problemas específicos da área de desenvolvimento de uma empresa nacional, os quais foram aferidos através de um diagnóstico composto pelo resultado de entrevistas realizadas a todos os seus membros e a alguns documentos quantitativos com dados referentes aos projetos realizados no último ano.

Este diagnóstico permitiu identificar que a gestão de projetos apresentava problemas nas suas várias fases e que não seguia nenhuma metodologia específica, embora tivesse uma base tradicional, e algum formalismo da fase de análise.

O diagnóstico identificou que a grande maioria dos projetos realizados nesta equipa são de muito curta duração (uma semana ou menos) e que incidem sobre um conjunto pequeno e limitado de plataformas tecnológicas. Embora estes projetos constituam um esforço muito pequeno, o facto de serem realizados pelo modelo em cascata, e de somarem tempos de análise e tempos de testes, leva a que os tempos totais de execução finais sejam muito superiores (valor médio de 58 dias úteis).

Estas duas características indicam que este tipo de projetos podem ser facilmente geridos através de metodologias ágeis, correspondendo as plataformas tecnológicas a produtos e os projetos a iterações, que podem ser enquadradas em conjuntos de funcionalidades maiores e transversais, as *releases*, de forma a tirar o melhor partido da equipa.

Este estudo concluiu também que a equipa não tinha a estrutura adequada para uma gestão ágil, tendo sido proposto um modelo alternativo

A adoção de um modelo ágil, para além de todas as vantagens inerentes à metodologia, permite dotar a gestão de ferramentas de grande visibilidade sobre os projetos. Através dos gráficos elaborados de forma quase automática sobre os dados do desempenho diário ou do quadro de tarefas, que deve preferencialmente ser físico e não implementado em *software*, é possível obter uma imagem imediata do estado atual do projeto e do trabalho que está por realizar.

A ênfase dada à comunicação, assim como o facto das técnicas estarem desenhadas para a incentivar, fazem com que estas equipas estejam muito mais informadas e motivadas. O

quadro de tarefas físico, colocado numa parede é um excelente exemplo de uma ferramenta que incentiva a comunicação, estando sempre presente durante a execução e nas reuniões diárias ou de ciclo. O exemplo do quadro mostra também o nível de visibilidade que se obtém com esta metodologia, tornando evidente para todos os membros da equipa o que está programado, o que está feito e o que resta fazer.

O resultado das entrevistas demostrou que muitos dos objetivos e necessidades manifestadas pelos membros da equipa correspondem a princípios das metodologias ágeis, pelo que é expectável que esta equipa atinja níveis de performance elevados com a adoção desta metodologia.

# 8 Possibilidades de trabalho futuro

A evolução mais natural deste trabalho, caso seja entendido pela ZTelc, é a implementação do *scrum* nesta equipa, o que requer alguma adaptação organizacional e o patrocínio da sua gestão, uma vez que muda radicalmente a forma como são abordados os projetos.

Se esta implementação for implementada de forma adequada, e uma vez que internamente quase todas as atividades são geridas como projetos, esta poderá ser uma metodologia a estender para outras áreas e outras equipas.

Este trabalho poderá também ser uma base de reflexão para que organizações com problemas semelhantes considerarem outra forma de realização dos seus projetos de desenvolvimento de software.

# 9 Bibliografia

Brooks, Frederick P. (1995), The Mythical Man-Month Essays on Software Engineering – Anniversary edition.

Chow, Tsun & Cao, Dac-Buu (2008) – A survey study of critical success factors in agile software projects. The Journal of Systems and Software 81, pp. 961-971

Cohn, Mike (2005), Agile Estimating and Planning. Massachusetts: Prentice Hall

Curtis, B., Hefley, W., Miller, S.(2001), People Maturity Capability Model, version 2.0, Pittsburgh: Software Engineering Institute, Carnegie Mellon University

Hughes, Bob, Cotterell Mike (1999), Software Project Management Second Edition. England: McGraw-Hill Publishing Company

Larman, C., Vodde B. (2008), Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale *Scrum*. Massachusetts: Addison-Wesley

Leffingwell, Dean (2011), Agile Software Requirements – Lean Requirements Practices for Teams, Programs, and the Enterprise. Massachusetts: Addison-Wesley

Management Consultancies Association (1993), Chief Executives views on project management performance.

Miguel, António (2010) , Gestão de projetos de software. 4ª Edição. Lisboa: FCA – Editora de Informática Lda.

Parkinson, C. Northcote (Novembro 1955) - Parkinson's Law. The Economist Newspaper

Project Management Institute (2008): A Guide to the Project Management Body of Knowledge (PMBOK Guide) 2008 Edition, Project Management Institute, Inc, Pennsylvania, USA

Rasmusson, Jonathan (2010), The Agile Samurai – How Agile Masters Deliver Great Software. Raleigh: The Pragmatic Bookshelf.

Runeson, P. & Host, M. (2009) – Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2), pp. 131-164

Santos, Mário (30 de abril de 2008), A entrevista semiestruturada, http://mariosantos700904.blogspot.pt/2008/04/entrevista-semi-estruturada.html

Schwaber, Ken & Sutherland, Jeff (2010) – Scrum: Developed and sustained. Scrum.org

Thamhain, H.J., Wilemon, D. L (1986), Criteria for controlling Software according to plan. Project Management Journal

Thayer, Pyster e Wood, "Major Issues in software engineering project management"- IEEE transactions on Software Engineering , volume 7, pag 333 – 342.

Turner, J. R. (1993), The Handbook of Project-Based Management. England: McGraw-Hill Publishing Company

Wateridge, John (1995), IT projects: a basis for success, International Journal of Project Management Vol. 13, No. 3, 169-172.

Wysocki, Robert K. (2003), Effective Project Management Traditional, Adaptive, Extreme – Third Edition. Wilwy Publishing

# 10 Anexo A – Questões das entrevistas semiestruturadas

Questões estipuladas para as entrevistas Semiestruturadas aos elementos da equipa de desenvolvimento da ZTelc

| **Questão** |
|---|
| **1 - Quais são as maiores dificuldades durante a fase de planeamento de um novo projeto?** |
| **2 - Quais são as maiores dificuldades durante o desenvolvimento de um projeto?** |
| **3 - Quais são os maiores problemas na gestão de recursos humanos da equipa?** |
| **4 - Que informação não tem e gostaria de ter de forma regular durante a execução de um projeto?** |
| **5 - Quais os principais pontos fortes da equipa?** |
| **6 - Quais os principais pontos fracos da equipa?** |
| **7 - Que sugestões apresentaria para melhorar a gestão de projetos nesta equipa?** |

# 11 Anexo B – Dicionário de termos ágeis[4]

| | |
|---|---|
| **Agile** | A conceptual framework for undertaking software projects. Agile methods are a family of development processes, not a single approach to software development |
| **Burndown Chart** | A burndown chart is a visual tool for measuring and displaying progress. Visually, a burndown chart is simply a line chart representing remaining work over time. Burndown charts are used to measure the progress of an agile project at both a iteration and project level. |
| **Daily Standup/Scrum** | A Daily Standup is a whole team meeting that happens at the same time every day that usually lasts 15 minutes or less. The meeting is designed to allow the entire team to synchronize with each other and to understand the flow and challenges of the development process. Each team member should provide the following information: what did I do yesterday, what am I planning to do today, and what impediments do I currently have? |
| **Done** | Also referred to as "Done Done", this term is used to describe all the various tasks that need to happen before a story is considered potentially releasable. |
| **Epic** | A very large user story that is eventually broken down into smaller stories. |
| **Estimation** | The process of agreeing on a size measurement for the stories, as well as the tasks required to implement those stories, in a product backlog. |
| **Kanban** | Kanban, pronounced /ˈkɑnˈbɑn/, is a method for developing products with an emphasis on just-in-time delivery and the optimization of flow of work on the team. It emphasizes that developers pull work from a queue, and the process, from definition of a task to its delivery to the customer, is displayed for participants to see. |
| **Lean** | Lean software development is a translation of Lean manufacturing and Lean IT principles and practices to the software development domain. Adapted from the Toyota Production System and is a set of techniques and principles for delivering more values with the same or less resources by eliminating waste across organizations and business processes |

[4] Retirado de http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

| | |
|---|---|
| **Pair Programming** | Pair programming is an agile software development technique in which two programmers work together at one workstation. One types in code while the other reviews each line of code as it is typed in. The person typing is called the driver. The person reviewing the code is called the observer (or navigator). The two programmers switch roles frequently. |
| **Planning Poker** | Also called Scrum poker, is a consensus-based technique for estimating, mostly used to estimate effort or relative size of tasks in software development . |
| **Product Backlog** | Acts as a repository for requirements targeted for release at some point. These are typically high level requirements with high level estimates provided by the product stakeholders. The requirements are listed on the backlog in priority order and maintained by the product owner. |
| **Product Owner** | The Product Owner represents the voice of the customer and is accountable for ensuring that the Team delivers value to the business. The Product Owner writes customer-centric items (typically user stories), prioritizes them, and adds them to the product backlog. Scrum teams should have one Product Owner. |
| **Retrospective** | A team meeting that happens at the end of every development iteration to review lessons learned and to discuss how the team can be more efficient in the future. It is based on the principles of applying the learning from the previous sprint to the upcoming sprint. |
| **Scrum** | Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. It is based on the adaptive and iterative methodology of software development. |
| **Scrum Master** | Scrum is accountable for removing impediments to the ability of the team to deliver the sprint goal/deliverables. The ScrumMaster is not the team leader but acts as a buffer between the team and any distracting influences. The ScrumMaster ensures that the Scrum process is used as intended. The ScrumMaster is the enforcer of rules. A key part of the ScrumMaster's role is to protect the team and keep them focused on the tasks at hand. The role has also been referred to as servant-leader to reinforce these dual perspectives. |
| **Spike** | A short, time-boxed piece of research, usually technical, on a single story that is intended to provide just enough information that the team can estimate the size of the story. |
| **Sprint/Iteration** | A fixed duration period of time where user stories are chosen to work on. The term Sprint comes from the Scrum methodology and |

| | |
|---|---|
| | is analogous to the term Iteration. A sprint is defined as a 2-4 week increment of software development activities that delivers working software and the end of the increment. External influences are not allowed to change the requirements of the stories being worked on. |
| **Sprint Backlog** | At the beginning of each sprint, the team has sprint planning with an end result being a backlog of work that the team anticipates completing at the end of the sprint. These are the items that the team will deliver against throughout the duration of the sprint. |
| **Sprint Planning** | Is a pre-sprint planning meeting attended by the core agile team. During the meeting the Product Owner describes the highest priority features to the team as described on the product backlog. The team then agrees on the number of features they can accomplish in the sprint and plans out the tasks required to achieve delivery of those features. The planning group works the features into User Stories and assigns Acceptance criteria to each story. |
| **Sprint Review** | Each Sprint is followed by a Sprint review. During this review the software developed in the previous Sprint is reviewed and if necessary new backlog items are added. |
| **Story Points** | Unit of estimation measuring complexity and size. |
| **Task** | A user story can be broken down in to one or more tasks. Tasks are estimated daily in hours (or story points) remaining by the developer working on them. |
| **Taskboard/Storyborad** | A wall chart with cards and sticky notes that represents all the work for in a given sprint. The notes are moved across the board to show progress. |
| **Team** | The Team is responsible for delivering the product. A Team is typically made up of 5–9 people with cross-functional skills who do the actual work (analyse, design, develop, test, technical communication, document, etc.). It is recommended that the Team be self-organizing and self-led, but often work with some form of project or team management. |
| **Test Driven Development** | Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, then produces code to pass that test and finally refactors the new code to acceptable standards. |
| **Timeboxing** | Timeboxing is a planning technique common in planning projects (typically for software development), where the schedule is divided into a number of separate time periods (timeboxes, normally two to six weeks long), with each part having its own |

| | |
|---|---|
| | deliverables, deadline and budget. |
| **User Stories** | A user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it. A user story is one or more sentences in the everyday or business language of the end user that captures what the user wants to achieve. A user story is also a placeholder for conversation between the users and the team. The user stories should be written by or for the customers for a software project and are their main instrument to influence the development of the software. User stories could also be written by developers to express non-functional requirements (security, performance, quality, etc.) |
| **Velocity** | It is a relative number which describes how much work the team can get done over a period of time. |
| **Vertical Slice** | Showing off a feature in an application that works from start to finish but may be limited in scope. For example a rope bridge crossing a chasm is immediately useful and allows people to cross. Having that in place can help to build a better bridge later. |
| **WIP** | Also known as Work in Progress is any work that has been started but has yet to be completed. |
| **XP** | A software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles (timeboxing), which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. |

# 12 Anexo C – Mapa de respostas às entrevistas de diagnóstico

## 12.1 Dificuldades na fase de Planeamento

| **Dificuldades da fase de planeamento** |
|---|
| Detetar atempadamente problemas que possam surgir na fase de desenvolvimento |
| Interpretações incorretas por parte do *Service Delivery* ou do cliente por desconhecimentos técnicos |
| Impossibilidade de comunicação direta entre a equipa de desenvolvimento e o cliente (a comunicação é feita exclusivamente através do *Service Delivery*, pelo que a comunicação é mais demorada) |
| O planeamento não tem um lugar próprio adequado, sendo normalmente realizado enquanto se está a executar outro projeto, não permitindo a calma e reflexão necessárias para a sua realização |
| Pouco empenho durante a realização do m- anual de projeto impedindo que neste sejam explicitados todos os requisitos de forma inequívoca e aceite por todos os *stakeholders*, obrigando a esclarecimentos e interrupções durante a fase de desenvolvimento |
| Os recursos humanos envolvidos no desenvolvimento são considerados apenas programadores e a sua opinião e conhecimento não são solicitados na análise do projeto (apenas na especificação técnica). Isto leva a situações incorretas na análise sendo criados requisitos inexatos condicionando um caminho que não seria o melhor |
| Pressão para que se assumam prazos irrealistas |
| Por vezes os requisitos são ambíguos e não inequívocos. Não existe uma linguagem comum entre o cliente, o *service delivery* e as várias subequipas de desenvolvimento, existindo interpretações diferentes e podendo resultar em última instância num projeto diferente do esperado pelo cliente |

## 12.2 Dificuldades nas fases de Execução, Monitorização e Controlo

| **Dificuldades durante as fases de Execução, monitorização e controlo** |
|---|
| Alterações aos requisitos durante o desenvolvimento |
| Dúvidas em relação a análise efetuada pelo *Service Delivery* |
| Interrupções constantes para resolver problemas noutros projetos já passados a produção |
| Alterações de prioridade regulares, obrigando as equipas a parar um projeto e pegar noutro. Esta situação tem impactos negativos sobre o projeto que fica parado, obrigando a tempo adicional no seu retorno |
| Relações entre as várias subequipas, que trabalham de forma isolada, cada uma com os seus projetos e prioridades, levando a períodos de espera quando um projeto de uma subequipa necessita de um componente de outra |
| As estimativas iniciais são uma previsão, mas existe uma pressão forte para cumprimento das "milestones" inicialmente definidas |
| Elevada pressão no cumprimento de prazos e de tarefas paralelas, prejudicando ou impedindo os testes e com prejuízos na qualidade das soluções desenvolvidas |
| Por vezes a análise do projeto não está fechada, podendo existir falhas de informação relevante e obrigando a voltar a tarefas de planeamento durante a fase de desenvolvimento |
| Os pedidos de alteração de requisitos por parte do cliente são comuns, não existindo uma metodologia definida para os tratar. Acabam por ser aceites com implicações nesse e nos outros projetos que estão a ser desenvolvidos pelas equipas |
| Muitas vezes o planeamento é deficiente ou inexistente, passando-se de imediato para a fase de desenvolvimento |
| Existem frequentemente dúvidas sobre os requisitos que levam a novas interações com o *Service Delivery* e com o cliente, dúvidas que deveriam ter sido esclarecidas anteriormente durante a análise e levantamento de requisitos |

## 12.3 Dificuldades na Gestão de Recursos Humanos

| **Dificuldades na Gestão de Recursos Humanos** |
| --- |
| Algumas pessoas estão associadas apenas a uma determinada ferramenta / plataforma não tendo abertura nem incentivo para colaborar noutro tipo de projetos |
| Falta de motivação e de identificação com os objetivos. Falta de sentimento de pertença |
| Não existe, durante o planeamento do projeto, uma correta definição dos seus intervenientes. É habitual que seja necessária a intervenção de elementos específicos pertencentes à equipa, que, deveria ser incluída no planeamento do projeto |
| O facto de não existir um gestor de projeto formal condiciona negativamente a coordenação de recursos |
| Os recursos humanos estão afetos a vários projetos em simultâneo e nem sempre são definidas prioridades e prazos credíveis para cada um deles |
| Equipas reais isoladas e de pequena dimensão |

## 12.4 Necessidades de Informação e Comunicação

| **Necessidades de Informação e Comunicação** |
| --- |
| Informação sobre o estado do projeto em que cada pessoa está envolvida e sobre os restantes projetos da equipa |
| Feedback do cliente da solução de forma a poder melhorar e desenvolver de forma mais ajustada às reais necessidade do cliente |
| Acesso a indicadores de importância relativa de cada projeto |
| Acesso a informação de dependências entre projetos e recursos |
| Visão em tempo real das atividades da equipa e possibilidade de comunicar / ajudar os vários elementos |
| Aumentar significativamente o nível de Informação e comunicação |
| Feedback das etapas já realizadas de acordo com o calendário acordado |
| Ponto de situação dos problemas encontrados de modo a que se possa agir antes de a meta temporal se atingida |
| Informação acerca dos projetos em curso e comunicação entre os diversos elementos das subequipas e dos seus múltiplos projetos. Uma vez que esta comunicação é deficiente ou inexistente, por vezes um novo projeto altera elementos que eram necessários para outro, criando uma situação de emergência que irá ter impacto em todos os projetos |
| Visibilidade global sobre o progresso do projeto e não apenas da micro tarefa em que cada programador está a trabalhar |

## 12.5 Identificação dos pontos fortes da equipa

| **Pontos fortes da Equipa** |
|---|
| Querer fazer bem e melhorar |
| Boa relação entre os elementos da equipa, com espirito de entreajuda |
| Boa capacidade de autoformação |
| Boa capacidade de adaptação |
| Conhecimentos técnicos muito abrangentes (mesmo não sendo de nível elevado) que permitem à equipa fazer muitas coisas diferentes |
| A flexibilidade e empenho pessoal dos membros da equipa permitem, geralmente, ultrapassar as dificuldades da falta de planeamento |

## 12.6 Identificação dos pontos fracos da equipa

| **Pontos fracos da Equipa** |
|---|
| Grande aversão ao risco, sendo esta condicionada pela administração da organização que tem demonstrado pouca tolerância a erros e falhas |
| Falta de comunicação entre as diversas subequipas |
| Falta de metodologias e de ferramentas para uma gestão eficiente dos projetos |
| Falta de rotatividade das pessoas pelas várias subequipas levando a uma acomodação |
| A organização “flat” da equipa não funciona, resultando em equipas pequenas, isoladas e sem informação |
| Não existe ninguém com o papel de gestor de projetos, cada subequipa é autónoma e a liderança não é clara |
| Tamanho reduzido de todas as subequipas e da equipa do ponto de vista global |

## 12.7 Identificação de sugestões por parte da equipa

| **Sugestões de melhoria para a gestão de projetos na equipa** |
|---|
| Criação efetiva do perfil de gestor de projetos |
| Identificação clara de hierarquias e responsabilidades (eventualmente redefinir as equipas) |
| Aumentar e melhorar a comunicação dentro da equipa |
| Investimento em formação tecnológica e de gestão de projetos |
| Visibilidade dos projetos e trabalhos em curso e de dados sobre a sua produtividade |
| Escudar os programadores para não poderem ser interrompidos durante um ciclo de desenvolvimento, permitindo-lhes dedicar-se totalmente a esse processo |
| Escolher uma metodologia de gestão de projetos e segui-la |
| Maior controlo entre as necessidades e os pedidos dos clientes: variadas vezes, por desinformação do cliente, este faz pedidos além das suas necessidades, aumentando o tempo de desenvolvimento, prejudicando projetos seguintes. É necessário que seja bem explícito o objetivo do cliente em cada pedido, e que haja abertura para uma análise critica por parte da equipa |
| Criar mecanismos de divulgação de informação. A informação deve chegar de igual modo a todos os elementos da equipa. Todos devem saber em que projetos a restante equipa está a trabalhar, de modo a criar as sinergias necessárias. Os projetos não são ilhas, e todos os elementos da equipa acabam por interagir uns com os outros |

# 13 Anexo D – Processos da fase de planeamento definidos pelo PMBOK

| Processo | Descrição e Observações |
|---|---|
| **Recolher os requisitos** | Definição e explicitação das necessidades dos *stakeholders* que devem ser endereçadas com este projeto. Segundo Miguel (2010) os requisitos devem cumprir um conjunto importante de critérios, antes de serem aceites para execução:<br>• Inequívocos (mensuráveis e testáveis)<br>• Auditáveis<br>• Completos<br>• Consistentes<br>• Aceitáveis<br>O PMBOK indica que que o sucesso do projeto está diretamente relacionado com o cuidado e atenção que é dada nesta fase à recolha e gestão dos requisitos do projeto e do produto a desenvolver. |
| **Definir o âmbito** | Descrição detalhada do projeto e do produto a obter com a sua execução. Miguel (2010) refere que esta descrição de âmbito é um trabalho dinâmico e que apenas estará completo na fase final do planeamento, quando são conhecidas todas as condições e limitações do projeto |
| **Criar a WBS** | Processo de subdivisão do trabalho do projeto em grupos mais pequenos e simples de gerir, e definição dos respetivos entregáveis |
| **Definição de atividades** | Definição concreta do conjunto de atividades necessárias para produzir cada um dos |

| Processo | Descrição e Observações |
|---|---|
| | entregáveis do projeto |
| **Sequenciação de atividades** | Identificação e documentação das relações entre as atividades do projeto |
| **Estimação dos recursos** | Estimativa dos tipos e quantidades de material, pessoas, equipamento ou outros elementos, necessários para a realização de cada atividade |
| **Estimação de tempos** | Estimativa das unidades de tempo necessárias para a realização de cada uma das atividades com os recursos previamente estimados.<br>Tal como refere Brooks (1995),<br>A maior causa de falha no cumprimento de projetos de desenvolvimento de *software*, maior que todas as outras juntas, é o cumprimento do prazo. Mas por que razão isto é tão comum?<br>Em primeiro lugar as técnicas de estimação estão pouco desenvolvidas, e partem de uma assunção irrealista de que no final tudo vai correr bem.<br>Em segundo lugar, as técnicas de estimação falaciosamente confundem esforço com progresso. |
| **Desenvolver o cronograma** | Criação de um cronograma a partir da análise da sequência de atividades, durações, necessidades de recursos e restrições de calendarização |
| **Estimação de custos** | Estimativa dos custos associados a cada recurso necessário para o atingimento das atividades do projeto |

| Processo | Descrição e Observações |
|---|---|
| **Determinação do orçamento** | Construção de um orçamento para o projeto através da agregação das estimativas de custo elaboradas para cada atividade individual ou pacote de trabalho |
| **Planeamento de qualidade** | Identificação dos requisitos de qualidade e / ou standards para ou projeto ou produto e documentação de como o projeto pode demonstrar a sua conformidade |
| **Desenvolvimento de plano de RH** | Identificação e documentação dos perfis necessários para o projeto, as suas responsabilidades e competências |
| **Planeamento de comunicações** | Determinação das necessidades de informação dos *stakeholders* e definição de um sistema de comunicação |
| **Planeamento da gestão de risco** | Definição de como é realizada a gestão de risco |
| **Identificação de riscos** | Identificação de riscos que possam afetar o projeto e documentação das suas características |
| **Análise qualitativa e quantitativa de riscos** | Priorização de riscos e identificação do seu impacto nas atividades do projeto |
| **Planeamento de respostas aos riscos** | Desenvolvimento de opções e ações que permitam explorar oportunidades e reduzir ameaças aos objetivos do projeto |

# 14 Anexo E – Processos da fase de execução definidos pelo PMBOK

| Processo | Descrição e observações |
|---|---|
| **Gerir e dirigir a execução do projeto** | Elaboração do trabalho definido no plano de projeto de forma a permitir o atingimento dos objetivos do projeto |
| **Garantir a garantia de qualidade** | Auditar os requisitos de qualidade e os resultados das medidas de controlo de qualidade para garantir que são seguidos os padrões de qualidade e definições operacionais |
| **Adquirir a equipa do projeto** | Confirmação da disponibilidade dos recursos humanos e obter a equipa necessária para realização das atividades do projeto |
| **Desenvolver a equipa do projeto** | Melhorar as competências da equipa, da sua interação e de toda a envolvência da equipa de forma a melhorar a sua performance no projeto |
| **Distribuição de informação** | Garantir que a informação relevante chega aos *stakeholders* |
| **Gerir as expectativas dos *stakeholders*** | Comunicar e trabalhar com os *stakeholders* para garantir que são atingidos os seus objetivos e que são de imediato tratadas quaisquer questões |
| **Gerir aquisições** | Gerir a escolha de fornecedores, a adjudicação do contrato e garantir que se obtém respostas atempadas por parte dos fornecedores |

# 15 Anexo F – Processos da fase de Monitorização e controlo definidos pelo PMBOK

| Processo | Descrição e Observações |
|---|---|
| Monitorizar e controlar o trabalho do projeto | Revisão e regulação do progresso do trabalho que está ser realizado para que sejam atingidos os objetivos definidos no plano de projeto. A monitorização inclui relatórios de ponto de situação, medidas de progresso e previsões de evolução. Os relatórios de performance possuem informação relacionada com o âmbito, calendarização, custos, recursos, qualidade e riscos. |
| Realizar o controlo de alterações | Revisão de todos os pedidos de alteração, aprovação e gestão de alterações no plano e nos entregáveis do projeto |
| Verificar o âmbito do projeto | Formalização de aceitação dos entregáveis do projeto que foram concluídos |
| Controlar o âmbito | Monitorização do estado do âmbito do projeto e do produto e gestão das alterações à *baseline* de âmbito |
| Controlar a calendarização | Monitorização do estado do projeto e atualização do seu progresso. Gestão de alterações à *baseline* de calendarização |
| Controlar os custos | Monitorização do estado do orçamento do projeto e gestão de alterações à *baseline* de custos |
| Realizar o controlo da qualidade | Monitorização e registo de resultados da execução das tarefas de qualidade para verificar a performance e proceder a recomendações de alteração caso sejam necessárias |

| Processo | Descrição e Observações |
|---|---|
| Reportar a performance | Recolher e distribuir informação de performance incluindo relatórios de estado, medidas de progresso e previsões acerca da evolução do projeto |
| Monitorizar e controlar os riscos | Implementação de planos de resposta a riscos, seguimento de riscos já identificados e monitorização de riscos residuais, identificação de novos riscos e avaliação da eficácia da gestão de riscos ao longo do projeto |
| Gerir aquisições | Gerir as relações com fornecedores, monitorização da performance do contrato e realização das alterações e correções necessárias |