Universidade de Aveiro
2012
Departamento de Economia, Gestão e Engenharia Industrial

DINA RAQUEL RODRIGUES RETROZ E SILVA

GESTÃO DE PROJETO DE SOFTWARE: CASO DE ESTUDO GREEN NA IUZ TECHNOLOGIES

Universidade de Aveiro
2012
Departamento de Economia, Gestão e Engenharia Industrial

DINA RAQUEL RODRIGUES RETROZ E SILVA

GESTÃO DE PROJETO DE SOFTWARE: CASO DE ESTUDO GREEN NA IUZ TECHNOLOGIES

Relatório de projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Gestão, realizada sob a orientação científica do Doutor Daniel Ferreira Polónia, Professor Auxiliar Convidado do Departamento de Economia, Gestão e Engenharia Industrial da Universidade de Aveiro

Dedico este trabalho à minha família por sempre me terem apoiado ao longo de todo o meu percurso académico.

o júri

presidente

Prof. Doutor Antonio Carrizo Moreira
Professor auxiliar da Universidade de Aveiro

Profª. Doutora Maria Madalena Gomes Vilas Boas
Professora auxiliar convidada da Universidade de Aveiro

Prof. Doutor Daniel Ferreira Polónia (orientador)
Professor auxiliar convidado da Universidade de Aveiro

agradecimentos

Agradeço à minha irmã, aos meus pais e aos meus amigos por todo o apoio e incentivo.
Agradeço aos meus orientadores pelo auxílio e orientação e a todos os meus colegas da *iUZ Technologies* pelo apoio ao longo de todo o estágio.

palavras-chave gestão de projeto, *software*, metodologia ágil, *scrum*, *user stories*.

resumo

Este documento relata o processo de estágio efetuado na *iUZ Technologies* entre Setembro de 2011 e Maio de 2012, onde a autora colaborou num processo de desenvolvimento de uma aplicação elaborada para um cliente da empresa de acolhimento.
Durante este trabalho, para além da gestão do projeto em causa, a autora envolveu-se no desenvolvimento do sistema de informação subjacente, desde o desenho dos requisitos até à validação e teste das aplicações desenvolvidas.
Para efetuar o trabalho foi necessário recorrer a múltiplas técnicas de planeamento e gestão de projeto, sendo efetuado neste relatório de estágio uma análise do estado da arte nesta área do conhecimento em termos de metodologias, que depois é confrontada com a realidade prática do dia-a-dia empresarial.
Para além disso, é também efetuada uma análise de metodologias de desenvolvimento e teste de sistemas de informação, sendo descrito o processo de desenvolvimento das aplicações com base em *user stories* e em metodologias ágeis de desenvolvimento de *software*.

keywords project management, software, agile methodologies, scrum, user stories

abstract

This document describes the traineeship process in iUZ Technologies between September 2011 and May 2012, where the author developed an application for a customer in the host company.
During this work, in addition to the management of project involved, the author became involved in the development of the underlying information system from the design of requirements until the validation and testing of applications developed.
To perform this work it was necessary to use multiple techniques for planning and project management, described in this report including an analysis of the state of the art in this area of knowledge in terms of methodologies, which is then faced with practical reality of day-to-day business.
In addition, it was also carried out an analysis of methodologies of development and testing of information systems, and described the process of developing applications based on user stories and in agile methodologies of software development.

Índice

Índice de Figuras

Índice de Quadros

Índice de Tabelas

Índice de Gráficos

1. Introdução

Este trabalho surgiu de uma proposta de plano de trabalho elaborada pela *iUZ Technologies* tendo em vista a participação num projeto de desenvolvimento de *software*. Nessa proposta constava um estágio curricular intitulado "Análise, especificação e verificação de implementação de requisitos de *software*" que permitiu a participação na gestão do projeto *Green*. O projeto *Green*, nome fictício atribuído por motivos de confidencialidade, iniciou-se a 03 de outubro de 2012 e na data de término de estágio, 25 de maio de 2012, continuava a decorrer.

Este relatório tem como principal objetivo a descrição e análise da metodologia utilizada, pela empresa *iUZ Technologies*, no desenvolvimento deste projeto.

O relatório está estruturado em 5 capítulos principais: introdução, enquadramento teórico, metodologias de trabalho, caso de estudo e conclusão, como se pode observar na Figura 1.

Figura 1 – Estrutura do relatório de estágio

O capítulo 2, enquadramento teórico, centra-se em 3 grandes temáticas: organização em equipas e em projeto, modelos de desenvolvimento de *software* e, análise e testes.

O enquadramento teórico principia com a definição e caraterização de conceitos aplicados a projetos do âmbito geral. Na organização em equipas e em projetos é definido o ciclo de vida de um projeto do âmbito geral, as diferentes fases de uma gestão de projetos e as várias arquiteturas organizacionais possíveis.

Em seguida, o enquadramento teórico apresentado é específico para projetos de desenvolvimento de *software*. Após diversas definições e a descrição das diversas fases e tarefas para o desenvolvimento de software, são apresentados diversas metodologias ágeis para o desenvolvimento de software, nomeadamente o método *scrum* utilizado na *iUZ Technologies*.

Na última temática abordada, descrevem-se de forma mais pormenorizada as fases de análise e testes do desenvolvimento de *software*, presentes na gestão de projeto de desenvolvimento de *software*. Para além da definição de vários conceitos inerentes a estas fases, discute-se qual a forma de realizar estas tarefas de forma a obter o máximo de produtividade com a melhor qualidade possível.

No capítulo 3 de metodologias de trabalho são discriminados todos os objetivos que se visam atingir com a realização do estágio curricular, bem como a previsão da sua distribuição temporal. Muito sucintamente, o estágio curricular tinha como objetivo adquirir mais conhecimentos sobre a atividade económica da empresa; compreender os procedimentos e metodologias utilizadas nos processos produtivos; participar em atividades de análise e especificação de requisitos de soluções a desenvolver, realizar tarefas integradas no processo produtivo da empresa como a verificação de requisitos e realização de testes de qualidade e de aceitação e; elaborar documentação de apoio aos utilizadores.

Posteriormente é apresentada a comparação entre os objetivos propostos e os objetivos atingidos, verificando-se alguma discrepância entre os mesmos.

O capítulo 4 é referente ao caso de estudo. Este capítulo inicia com uma descrição da empresa *iUZ Technologies* apresentando-se uma caraterização dos produtos e serviços disponibilizados pela empresa. Para além desta informação é apresentada a estrutura organizacional da iUZ e apresentada uma descrição do projeto “Green”, projeto sobre o qual foi analisada a metodologia aplicada na empresa em questão. Finalmente, chegamos à operacionalização do trabalho em que são descritas e analisadas as formas e metodologias de desenvolvimento deste projeto.

O relatório encerra com as conclusões do mesmo em que são mencionadas algumas limitações do relatório e propostas para trabalhos futuros, bem como uma análise geral do estágio.

Em anexo encontra-se uma descrição sumária das atividades desenvolvidas ao longo do estágio (Anexo 1) e diagramas de atividade elaborados no decorrer do estágio. No anexo 3 consta um posfácio realizado pela empresa acolhedora, intitulado de carta de recomendação.

2. Enquadramento teórico

O enquadramento teórico deste relatório encontra-se focado em três grandes temáticas, iniciando a exposição pela organização em equipas e em projeto, seguido de modelos de desenvolvimento de *software* e terminando com a análise e testes que se divide em análise e verificação e, testes e verificação (Figura 2).

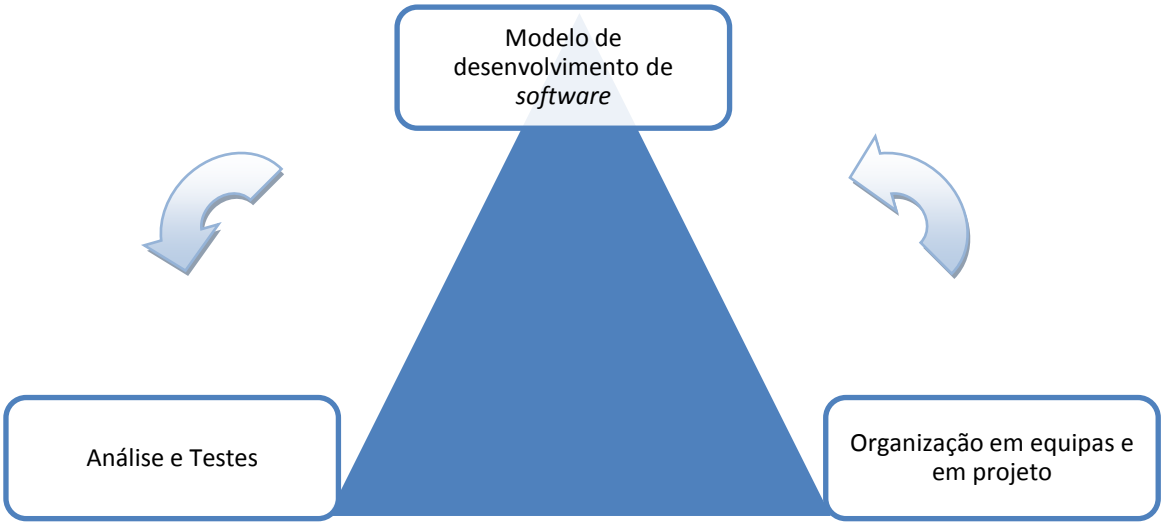


Figura 2 – Enquadramento teórico do relatório de estágio

2.1. Organização em equipas e em projeto

Nas últimas décadas, o mercado tem sido marcado pelo rápido crescimento do uso de gestão de projeto como um meio das empresas alcançarem os seus objetivos.

Por projeto pode-se entender um conjunto de atividades e tarefas não repetitivas e planificadas, sequencialmente independentes, complexas e interligadas, realizadas de acordo com especificações técnicas pré-determinadas e visa atingir objetivos sob especificações pré-fixadas de custos e prazos. Existem várias definições de projeto, das quais destacamos as seguintes: segundo o PMI (*Project Management Institute*) projeto define-se como esforço temporário realizado para criar um produto ou serviço, únicos (Meredith & Mantel, 2010); segundo Duncan (2008) define-se como uma tentativa de criar um produto, serviço e/ou resultado, com enquadramento temporal, o que significa que o mesmo possui um princípio e um fim.

Um projeto pode ter dimensões muito variadas, podendo ser realizado por uma única pessoa, por uma equipa de uma empresa ou por várias equipas de diversas empresas. A área em que está inserido o projeto também pode ser muito distinta, nomeadamente, do âmbito económico, ambiental e social.

Apresenta-se, de seguida, o ciclo de vida de um projeto, as várias fases da gestão do projeto e as diversas formas de arquitetura organizacional.

2.1.1. Ciclo de vida de um projeto

As organizações que desenvolvem projetos habitualmente estruturam-no em várias fases tendo como objetivo assegurar um controlo mais eficiente e mais eficaz da sua gestão e proporcionar uma ligação harmoniosa entre o projeto e a atividade operacional (Duncan, 2008).

O conjunto das fases do projeto constitui o seu ciclo de vida. Segundo Meredith e Mantel (2010), por ciclo de vida entende-se um conceito padrão de produto ou projeto que passa através de uma fase inicial, fase de construção fase de maturação e fase terminal.

Cada fase do projeto é caraterizada pela conclusão de uma, ou mais, atividades, sendo estas entendidas como um produto tangível, verificável do trabalho como, por exemplo, um protótipo (Duncan, 2008).

As atividades e as fases são constituintes de uma sequência lógica concebida para assegurar uma definição adequada do "produto final" do projeto. Normalmente, após a conclusão de cada fase do projeto, realiza-se uma revisão das principais atividades da performance do projeto, denominados *milestones*. Esta revisão é realizada com o propósito de avaliar a continuidade do projeto e identificar e corrigir eventuais desvios.

O ciclo de vida do projeto permite definir o início e o fim do projeto, as atividades e tarefas que deverão estar incluídas em cada fase do projeto e, ainda, as pessoas que devem estar envolvidas em cada fase do mesmo. Nas fases iniciais do projeto verifica-se uma grande influência dos diferentes interessados no mesmo (designados por *stakeholders*), quer ao nível da definição das suas caraterísticas, quer ao nível do preço. Esta influência vai-se reduzindo progressivamente com o decorrer do projeto e aproximação da sua conclusão, uma vez que, as alterações do mesmo, têm sido introduzidas ao longo da sua realização.

O ciclo de vida do projeto numa organização pode ser distinto do praticado em outra organização. Pode aparecer uma grande variedade de nomes para funções semelhantes e percursos diferentes para a realização de um projeto, abrangendo uma grande variedade no número de fases. No entanto, iremos considerar as seguintes fases principais, que se encontram discriminadas e detalhadas na Figura 3 (Duncan, 2008):

- Iniciação/ Preparação / Avaliação
- Planeamento / Execução
- Exploração
- Encerramento.

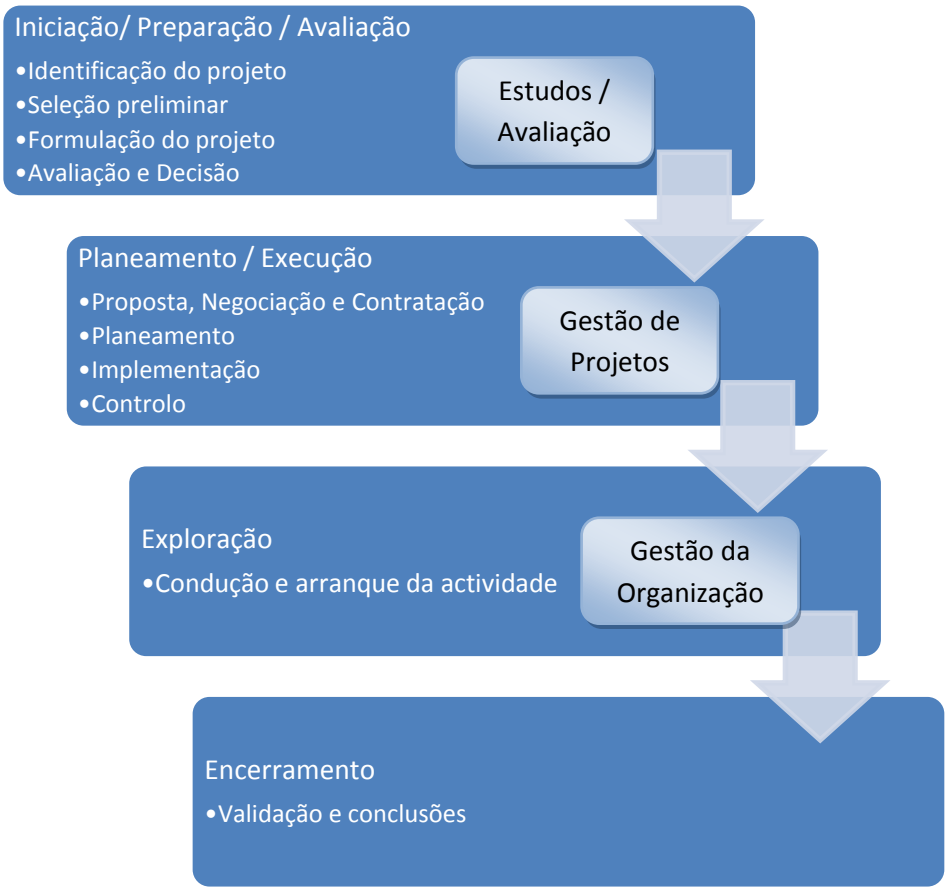


Figura 3 – Ciclo de vida de um projeto

Para transmitir a diversidade de ciclos de vida existentes na elaboração de projetos, mostramos na Figura 4 outro ciclo de vida. O exemplo apresentado é relativo à Microsoft (MOF - Microsoft Operations Framework) e aplicado às Tecnologias de Informação (TI) (Microsoft, 2008).

Neste caso, temos as seguintes fases:

- Fase de Planeamento: geralmente a fase preliminar, onde o objetivo é planear e otimizar uma estratégia de serviços de TI, a fim de apoiar as metas e objetivos de negócios.
- Fase de Entrega: o objetivo é garantir que os serviços de TI são desenvolvidas de forma eficaz, são implantados com sucesso, e estão prontos para a fase de Operações.
- Fase de Operação: o objetivo desta fase é garantir que os serviços de TI são verificados, mantidos e suportados de forma que atenda às necessidades e expectativas do negócio.
- Gestão: base do ciclo de vida de serviços de TI cujo objetivo é proporcionar os princípios de operacionalização e melhores práticas para assegurar que o investimento em TI agregue valor esperado aos negócios num nível aceitável de risco. Nesta fase tem-se atenção à gestão de TI, ao risco, cumprimento de objetivos, papéis

e responsabilidades, mudanças e configuração, sendo estes processos presentes em todas as fases do ciclo de vida.

Figura 4 – Ciclo de Vida de TI (Microsoft, 2008)

Ao longo do ciclo de vida do projeto, os custos e o nível de pessoal necessário para a realização do mesmo vai-se alterando, sendo a execução do projeto a fase que consome mais recursos humanos e financeiros, como o ilustrado na Figura 5.

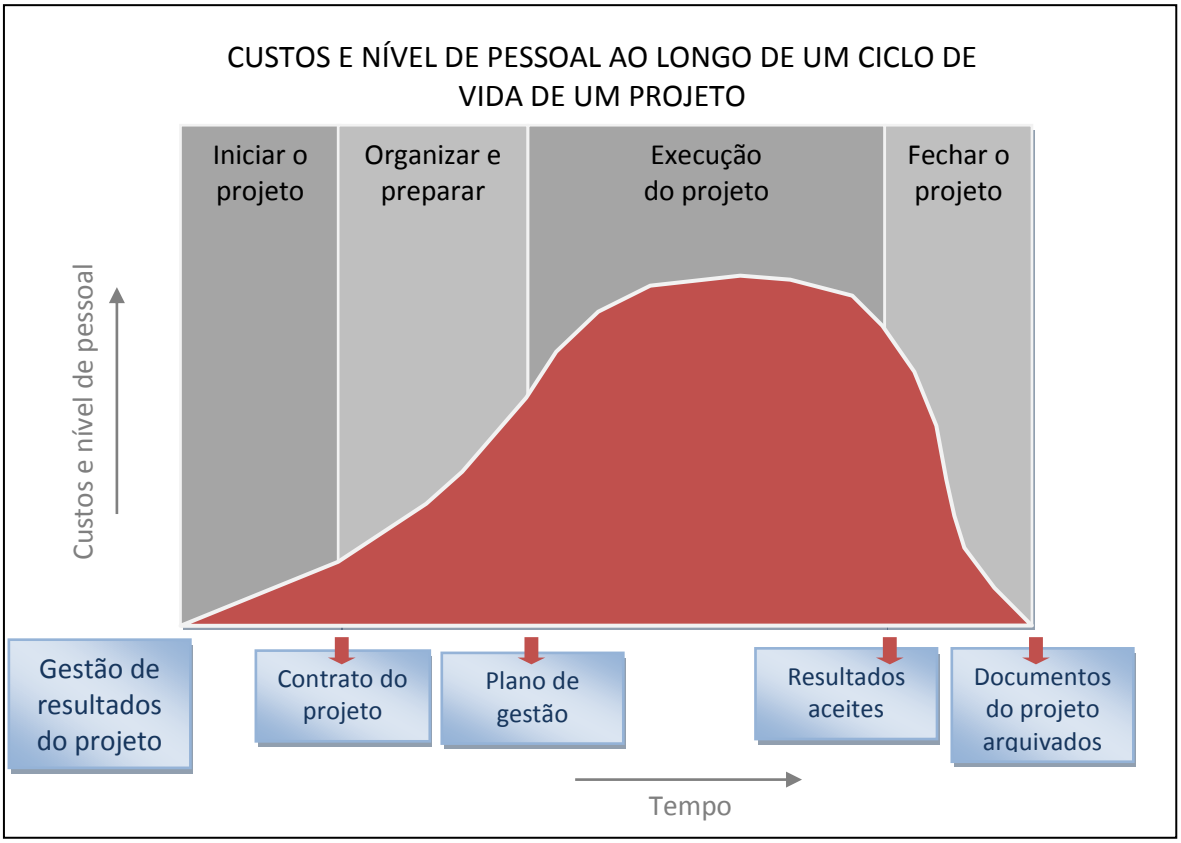


Figura 5 – Custos e nível de pessoal ao longo de um ciclo de vida de um projeto típico, adaptado de PMBOK (Duncan, 2008)

Apesar de todas as etapas descritas e planeamento existente, existem riscos durante o ciclo de vida do projeto. Existe um grande conforto quando é possível prever com certeza quando começar o projeto, como o realizar, qual o tempo disponível e os seus custos. Em alguns casos, a experiência em construção de projetos gera umas previsões com uma precisão razoável mas frequentemente não é possível. Deste facto resulta uma considerável incerteza sobre a capacidade de satisfazer os objetivos do projeto.

A incerteza aparece desde o início do projeto e vai aumentando até à conclusão do mesmo. No desenvolvimento atual de projetos, o grau de incerteza relativo ao produto final está reduzido. É comum estabelecer previsões de desenvolvimento, tempo e custos em intervalos fixos do ciclo de vida do projeto ou quando metas tecnológicas específicas são atingidas. Em qualquer evento, há medida que o projeto é desenvolvido, menos incerteza existe relativa a alcançar o objetivo final (Meredith & Mantel, 2010).

Existe uma relação entre o tempo despendido e os custos do projeto (Figura 5). Lidar com a incerteza que existe a envolver esta relação é da responsabilidade do gestor de projeto.

Em seguida apresentam-se mais atividades inerentes ao gestor de projeto e uma descrição detalhada sobre a gestão de projeto.

2.1.2. Gestão de Projetos

Para realizar o planeamento e execução de projetos, chegamos a um novo conceito, que está presente na Figura 3, denominado por gestão de projetos. A gestão de projetos é constituída por meios, técnicas e conceitos utilizados para desenvolver os projetos e alcançar os seus objetivos (Meredith & Mantel, 2010).

A gestão de projetos é acompanhada pela aplicação e integração de 42 grupos lógicos de processos comprimidos em 5 grupos de processos (ver

Tabela 1 e Figura 6) (Duncan, 2008; Haughey, n. d.; Miqdadi, n. d.) os quais se distinguem em:

- Iniciação;
- Planeamento;
- Execução;
- Monotorização e Controlo;
- Encerramento.

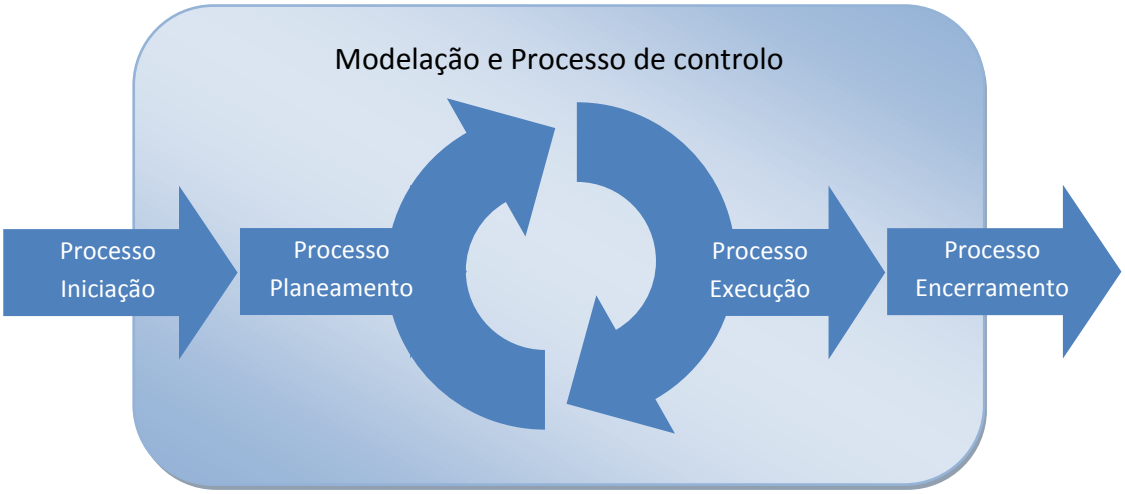


Figura 6 – Exemplo das fases de um projeto, adaptado de PMBOK (Duncan, 2008)

Um modelo de gestão de projetos permite contribuir para estruturar, metódica e progressivamente, a realidade futura da organização e aferir um maior rigor na afetação de tarefas e simplificação de controlo. Um projeto é caraterizado por ter uma duração limitada, quer seja estipulado o limite internamente pela gestão ou pelo cliente, e por dispor de recursos humanos, financeiros e técnicos também limitados. Estas caraterísticas conduzem a uma missão distintiva e a uma conclusão bem definida. No entanto, a aleatoriedade e o risco são outras caraterísticas bem presentes em todos os projetos (Duncan, 2008).

A gestão de um projeto típica inclui:

- Identificação de requisitos;
- Registar as várias necessidades, compromissos e expectativas dos *stakeholders* no planeamento do projeto;
- Realizar o balanço do projeto competitivo registando:
    - Âmbito, Qualidade, Prazos, Orçamento, Recursos e Risco (Duncan, 2008).

A especificidade do projeto vai influenciar as restrições em que o gestor de projeto necessita de se focar. Estes fatores estão todos relacionados e quando se verificam alterações num deles, pelo menos um dos outros é afetado. A equipa do projeto deve ter capacidade aceder à situação e de gerir as alterações de forma a obter um projeto com sucesso.

Por causa do potencial da mudança, o plano do gestor de projeto é interativo e é elaborado progressivamente ao longo do ciclo de vida. A elaboração progressiva envolve uma atualização constante e um plano mais detalhado, com informação específica e, estimativas mais precisas ficam disponíveis. A elaboração de um projeto também permite à equipa aceder a um maior nível de detalhe à medida que o projeto evolui.

Áreas do conhecimento	Grupos do Processo de Gestão do Projeto				
	Iniciação	Planeamento	Execução	Monotorização e Controlo	Encerramento
Gestão da Integração do Projeto	Contrato do desenvolvimento do projeto	Desenvolver plano de gestão de projeto	Orientar e gerir o projeto de execução	Monitorizar e controlar o trabalho do projeto Realizar o controlo integrado de alterações	Encerrar o projeto ou fase
Gestão do Âmbito do Projeto		Recolher requisitos Definir âmbito Criar WBS[1]		Verificar âmbito Controlar âmbito	
Gestão do Tempo do Projeto		Definir atividades Ordenar atividades Estimar recursos de atividades Estimar duração de atividades Desenvolver calendário		Controlar calendário	
Gestão do Custo do Projeto		Estimar custos Determinar orçamento		Controlar os custos	
Gestão da Qualidade do Projeto		Plano de qualidade	Executar garantia de qualidade	Realizar controlo de qualidade	
Gestão dos Recursos Humanos do Projeto		Plano de desenvolvimento de recursos humanos	Contratar equipa de projeto Desenvolver equipa de projeto Gerir equipa de projeto		
Gestão das Comunicações do Projeto	Identificar *stakeholders*	Plano de comunicações	Distribuir informações Gestão das expectativas dos *stakeholders*	Relatório de desempenho	
Gestão do Risco do Projeto		Gestão do plano de riscos Identificar riscos Realizar análise qualitativa do risco Realizar análise quantitativa do risco Plano de resposta ao risco		Monotorização e controlo de riscos	
Gestão de Contratos de Projeto		Plano de contratos	Condução de contratos	Administração de contratos	Fechar contratos

Tabela 1 – Grupos do Processo de Gestão do Projeto, (*PMBOK Guide 4th Edition Process Groups Knowledge Areas Mapping*)

[1] WBS (Work Breakdown Structure): Processo de divisão do projeto em etapas com componentes de trabalho menores e mais executáveis (Miqdadi, n. d.)

Apesar da variedade que se pode encontrar numa gestão de projetos, a estrutura organizacional inerente ao projeto também pode ser diversa, como se apresenta em seguida.

2.1.3. Arquitetura organizacional

Um projeto pode ser realizado por diversos intervenientes como, por exemplo, o gestor do projeto, o cliente, que será o indivíduo ou organização que irá usar o produto do projeto, a organização que desenvolve o projeto, a equipa de projeto, a equipa de gestão de projeto e os fornecedores.

A realização de um projeto é influenciada pela cultura organizacional, o estilo e estrutura do mesmo. No entanto, estes não são os únicos fatores que influenciam a execução do projeto sendo o grau de maturidade da gestão do projeto e o seu sistema de gestão bastante relevantes, bem como, entidades externas no caso de realizar parcerias ou *joint ventures*.

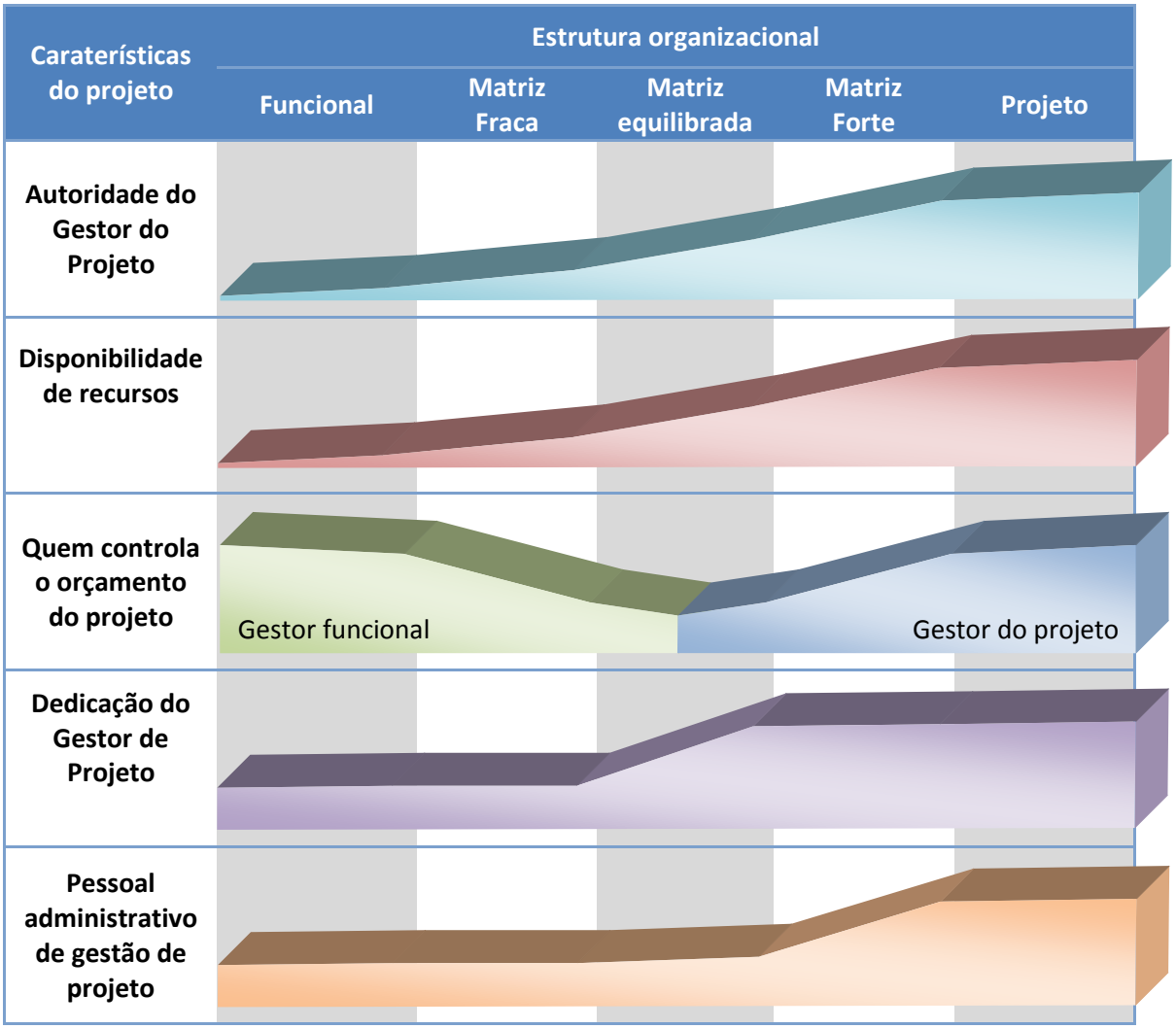


Tabela 2 – Influências organizacionais por projeto, adaptado de PMBOK ((Duncan, 2008))

Duncan (2008) considera a cultura organizacional e o estilo como os fatores com maior peso na realização do projeto. A cultura organizacional engloba o conhecimento partilhado relativo às normas praticadas dentro da empresa, ao modo de realizar o trabalho e que meios consideram adequados para a realização do projeto. A cultura organizacional pode conter outros fatores como, por exemplo, crenças, valores, visão, normas e políticas (Stair & Reynolds, 2005). Devido à sua grande abrangência de fatores, muitas organizações desenvolvem uma cultura organizacional própria que se tenta adaptar, o melhor possível às suas necessidades.

A estrutura organizacional permite organizar as operações, tarefas, processos de desempenho, de informação, de entidade e redes de comunicação. A estrutura organizacional permite estabelecer a hierarquia, agregar e definir as relações entre os serviços, os canais de comunicação, pode afetar a avaliação de recursos e influência a forma como os projetos são conduzidos (Coelho, Lisboa, Coelho, & Almeida, 2004; Karimi & Noori, 2011). A estrutura organizacional pode ser funcional, por matriz (distinguindo-se três níveis, fraca, equilibrada e forte), ou por projeto. Em seguida serão apresentados vários tipos de estruturas organizacionais com as suas caraterísticas principais.

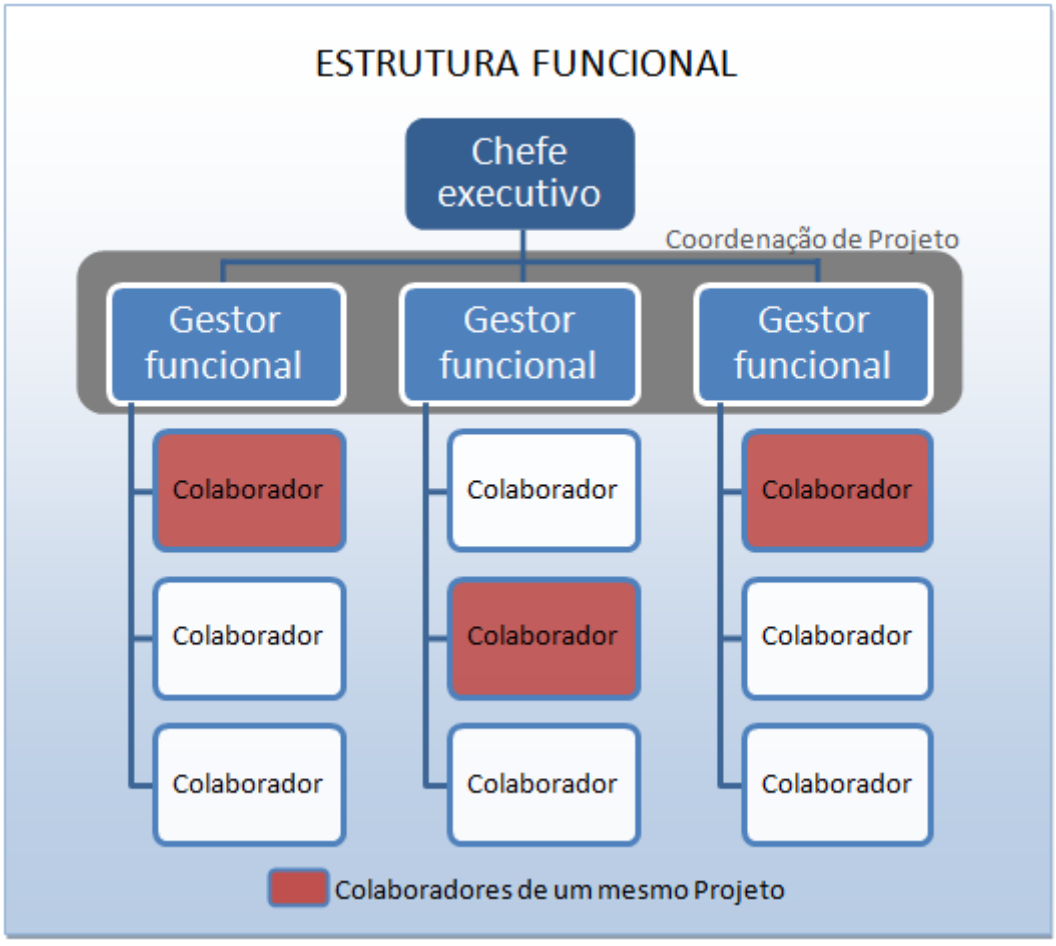


Figura 7 – Estrutura Organizacional Funcional, adaptado de PMBOK (Duncan, 2008)

A estrutura funcional, ilustrada na Figura 7, possui uma hierarquia onde cada colaborador tem um superior hierárquico claramente determinado. Neste modelo clássico, os trabalhadores são agrupados por especialidade como produção, marketing, engenharia e administração. O controlo do projeto é da responsabilidade do gestor funcional e a autoridade do gestor do projeto é reduzida ou inexistente.

Este modelo possui diversas vantagens, nomeadamente, uma hierarquia clara, precisa e com linhas de responsabilidade bem definidas, permite um controlo adequado do orçamento, a especialização dos colaboradores e é aconselhado a empresas que pretendem realizar produção em massa. A sua estrutura é extremamente simples e intuitiva. Ao permitir agrupar especialidades e competências, promove o aproveitamento de sinergias, duplicação de recursos e redução dos custos globais de funcionamento (Coelho, et al., 2004; Duncan, 2008).

Por outro lado, possui uma organização burocratizada, valoriza a estabilidade e inércia comportamental e possui uma coordenação horizontal complexa que exige tempo, o que é negativo para a realização do projeto, podendo apresentar problemas de coordenação e comunicação, tornar-se lenta e pouco fiável. Juntamente com estes motivos, a baixa motivação dos colaboradores e inovação da empresa e a grande dependência de subcontratados e consultores, no respeitante ao conhecimento sobre gestão do projeto constituem as desvantagens mais relevantes da aplicação da estrutura funcional.

Este tipo de estrutura pode ser difícil de adaptar a contextos dinâmicos que se vivem na atualidade, combinado melhor com ambientes estáveis (Coelho, et al., 2004).

A estrutura organizacional por matriz, também denominada estrutura organizacional multidimensional, é uma estrutura que está entre a estrutura funcional e a estrutura por projeto. Este mdelo pode incorporar diversas estruturas ao mesmo tempo (Stair & Reynolds, 2005).

Esta estrutura matricial pode ser classificada em fraca, representada na Figura 8, equilibrada caraterizada na Figura 10 e forte, apresentada na Figura 9. A estrutura matricial fraca é semelhante à estrutura funcional e a estrutura matricial forte aproxima-se da estrutura por projeto. A maior diferença entre estes três níveis de estrutura matricial é referente à

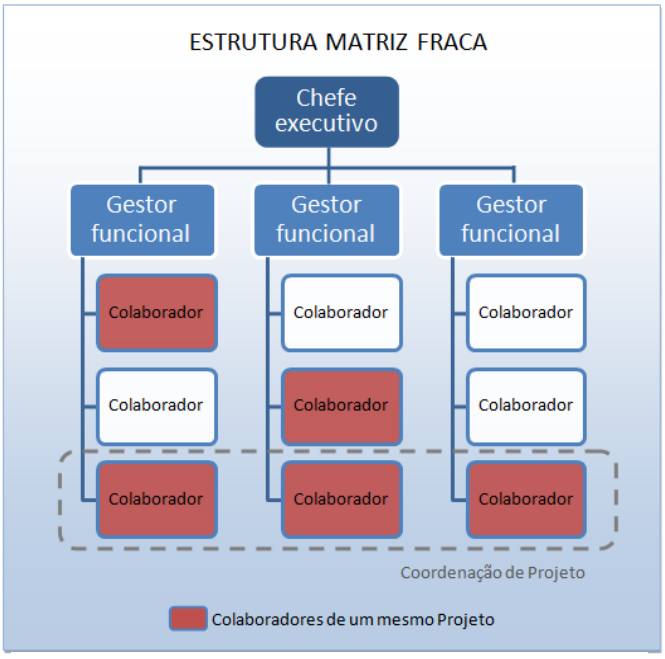


Figura 8 – Estrutura Organizacional Matriz Fraca, adaptada de PMBOK (Duncan, 2008)

autoridade do gestor de projeto que passa de reduzida, na matriz fraca, para moderada, na matriz equilibrada e de moderada a alta na matriz forte. O controlo dos recursos do projeto passa do gestor funcional da matriz fraca para o gestor do projeto na matriz forte.

Estas estruturas apresentam canais de informação horizontais e verticais, tendo a linha horizontal a capacidade para implementar o projeto por si e a linha vertical ser utilizada para efeitos administrativos. No entanto, o facto de possuir estas duas linhas pode originar tensões entre estas. Nestes modelos, os especialistas podem ser trocados entre os diversos projetos o que proporciona uma minimização de custos, é possível uma resposta rápida e alterações do meio envolvente e a autoridade e a responsabilidade são participadas.

Do ponto de vista conceptual, a estrutura matricial aproxima-se do que poderia ser a estrutura "Ideal". Nestas estruturas estão ausentes os pontos fracos das outras estruturas: a segmentação de interesses e objetivos da estrutura funcional foi colmatada pela coordenação horizontal e a multiplicação de recursos da estrutura por projeto foram reduzidos à sua expressão mínima. O resultado final parece ser uma estrutura ágil, adaptativa, capaz de maximizar a eficiência interna e de a combinar com uma grande capacidade de alinhamento com os fatores externos

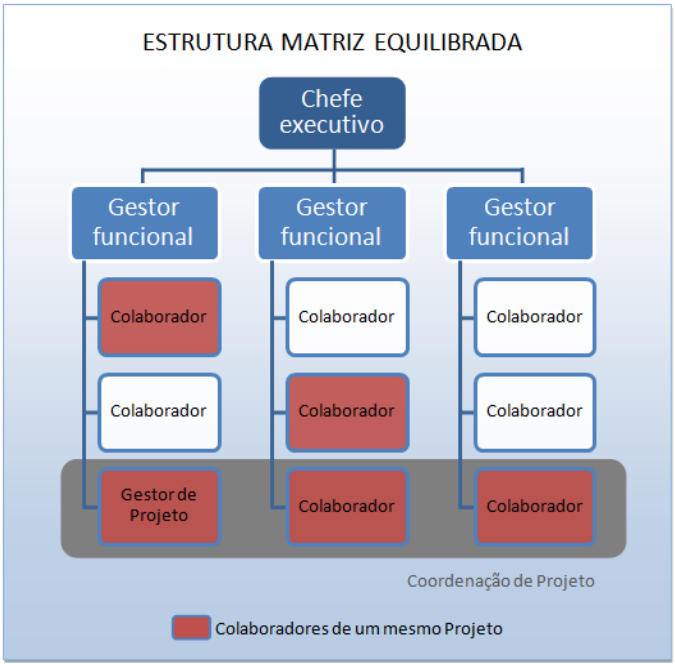


Figura 10 – Estrutura Organizacional Matriz Equilibrada, adaptada de PMBOK (Duncan, 2008)

Figura 9 – Estrutura Organizacional Matriz Forte, adaptada de PMBOK (Duncan, 2008)

(Coelho, et al., 2004).

Todavia, estas estruturas apresentam algumas desvantagens, nomeadamente, múltiplas linhas de autoridade, a tendência para alteração constante das prioridades, conflitos constantes, duplicação de esforços e um grande esforço inicial na definição de objetivos e procedimentos, estando muita gente envolvida na tomada de decisão (Coelho, et al., 2004; Stair & Reynolds, 2005).

No modelo orientado por projeto, retratado na Figura 11, verificamos a existência de uma equipa destinada ao desenvolvimento de um projeto controlado pelo gestor do projeto sendo a sua autoridade muito elevada ou mesmo total. Este modelo é aplicado em situações que pretendem o desenvolvimento de projetos ou serviços únicos e diferenciados como, por exemplo, o caso de projetos de arquitetura ou de engenharia de sistemas, em que são desenvolvidos produtos exclusivos para cada cliente.

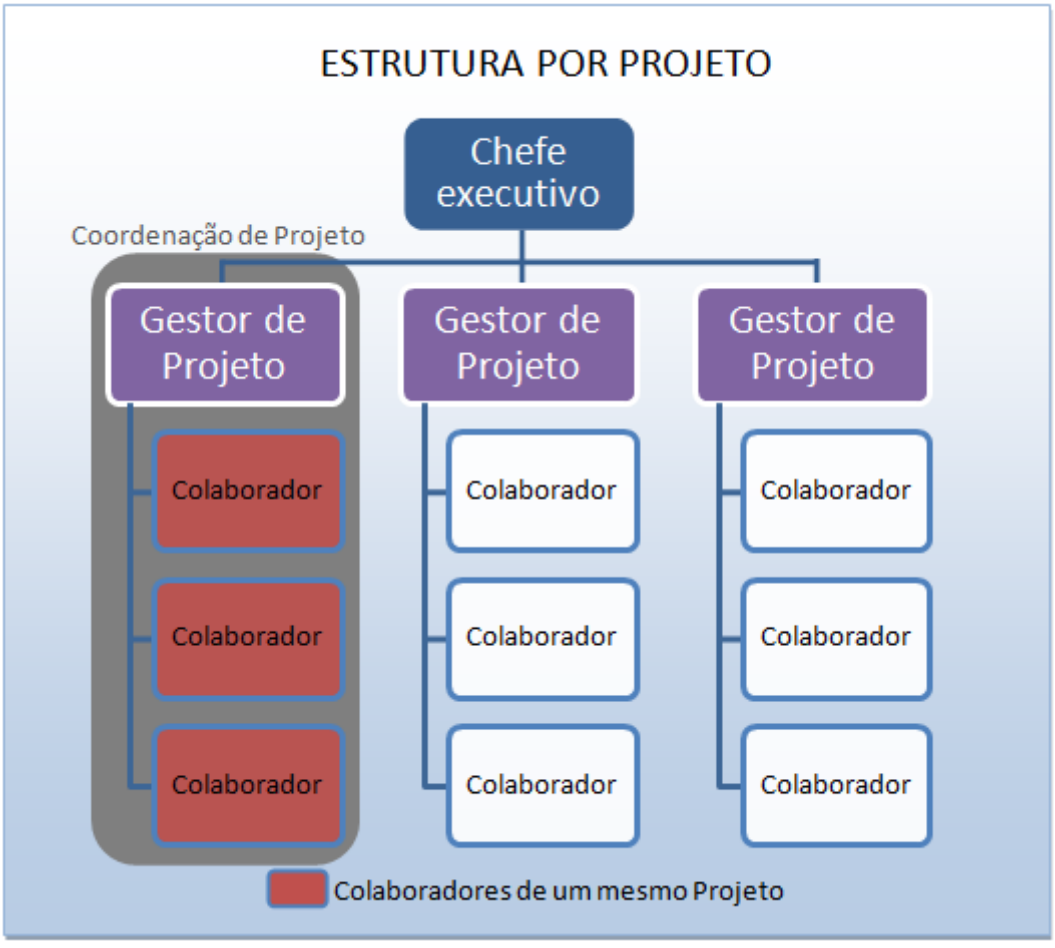


Figura 11 – Estrutura Organizacional por Projeto, adaptada de PMBOK (Duncan, 2008)

Neste caso, a gestão da empresa fica disponível para tarefas diferentes da gestão de projeto. Na gestão está definida claramente uma autoridade linear e precisa relativamente ao projeto e permite flexibilidade na fixação de tempos e de custos, constituindo as vantagens da aplicação deste modelo. Este modelo também evita a segmentação de interesses e a possibilidade de conflito de objetivos, aumentando a capacidade de gerar respostas imediatas, criativas e inovadoras, estando melhor preparados para responder a ambientes dinâmicos.

Como desvantagem encontra-se a possibilidade de inexistência da continuidade na carreira profissional dos colaboradores com a finalização do projeto, a dependência e ligação com o projeto após as atividades terminarem e o custo duplicado da estrutura funcional que possuirá técnicos com a mesma especialização no âmbito da empresa. Ao separar os especialistas por divisões, perdem-se as sinergias que poderiam resultar da sua agregação num só departamento (Coelho, et al., 2004)

Ao longo desta secção de organização em equipas e em projeto foi descrito um ciclo de vida de um projeto, especificando as várias etapas que este contém, nomeadamente a gestão de projetos, que é abordada com mais detalhe na secção seguinte.

Para finalizar a organização em equipas e em projeto foram apresentadas as várias formas de organizar as equipas, desde a funcional até à de projeto.

Na secção seguinte será abordada a gestão de projeto aplicada ao desenvolvimento de *software*.

2.2. Gestão de Projeto de desenvolvimento de software

Ao longo dos tempos, o processo de desenvolvimento de *software* foi-se alterando. Por *software* entende-se o "conjunto de programas, processos e regras, e, eventualmente, de documentação, relativos ao funcionamento de um conjunto de tratamento da informação" (Priberam, 2010). Apesar de alguns autores utilizarem *software* e sistemas de informação indistintamente, na realidade, sistemas de informação tem um significado mais abrangente.

Segundo Stair e Reynolds (Stair & Reynolds, 2005), sistemas de informação são conjuntos de elementos inter-relacionados que reúnem (entrada), manipulam e armazenam (processamento) e disseminam (saídas) dados e informação. A entrada é a atividade de captura e reunião de novos dados, o processamento envolve a conversão ou transformação de dados em saídas úteis e a saída envolve a produção de informações úteis. O *feedback* é a saída usada para ajustar ou alterar a entrada ou o processamento (Figura 12). Os componentes de um sistema de informação computacional incluem *hardware*, *software*, base de dados, telecomunicações e internet, pessoas e procedimentos.

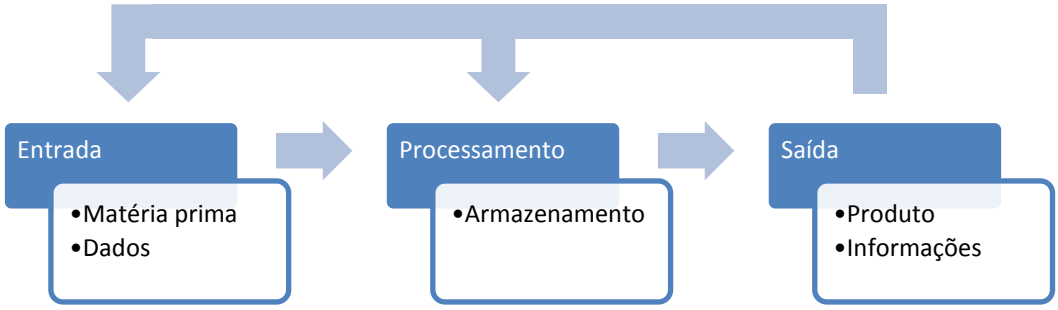


Figura 12 – Componentes de um Sistema de Informação (adaptado de Stair & Reynolds, 2005)

Uma das ideias dominantes de para a melhoria de desenvolvimento de *software* é a necessidade de aplicar um processo com fases bem definidas, que se dividem em conceitos mais elementares (tarefas e atividades). Segundo Silva e Videira (2001) as três grandes fases do processo (Figura 13) são:

- Concepção;
- Implementação;
- Manutenção.

Existem outros autores que consideram que o desenvolvimento também é uma fase e, por isso, existem 4 fases principais: concepção, desenvolvimento, implementação e manutenção (Gallegos, Senft, Manson, & Gonzales, 2004).

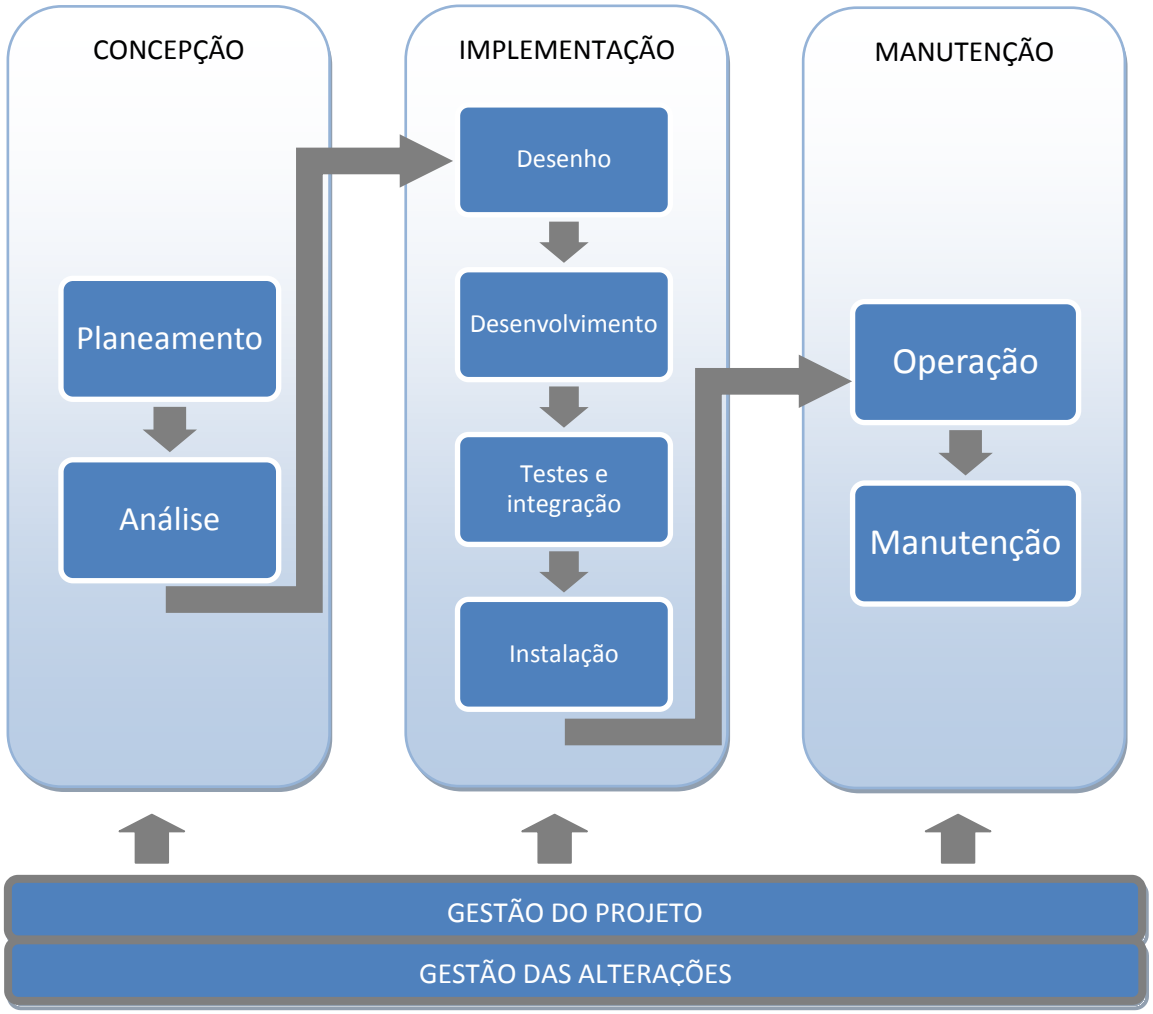


Figura 13 – Fases e tarefas do processo do desenvolvimento de software, (adaptado de Silva & Videira, 2001)

Regressando ao modelo anterior de 3 fases, a concepção tem como objetivo identificar "o que é que o sistema deve fazer", ou seja, deve identificar a informação que deve processar, as funcionalidades a implementar, as restrições existentes e os critérios que determinam o sucesso e a aceitação. Para além destes fatores, devem ser estudadas e avaliadas diferentes alternativas e efetuada a respetiva seleção.

Na implementação o propósito é identificar "o como fazer o sistema" e construí-lo realmente. Nesta fase serão definidas e construídas as estruturas de dados, os programas, os módulos, as interfaces internas e externas e os testes a realizar. No final desta fase, deverá estar disponibilizado o sistema de forma funcional.

A manutenção inclui todas as alterações posteriores à aceitação do produto pelo cliente final. Nestas alterações pode-se encontrar a correção de erros, a introdução de melhorias e novas funcionalidades.

Estas fases e tarefas, acima descritas, podem ser concretizadas em diferentes modelos os quais são caraterizados pela sequência das tarefas e pela diferente especificidade de fases e tarefas.

Os modelos de desenvolvimento de *software* têm como objetivo a redução de riscos do desenvolvimento de *software*. Existem diversos modelos criados, desde os modelos de desenvolvimento clássico como, por exemplo, o modelo em cascata, o modelo em V e modelo de prototipagem, até os modelos de desenvolvimento evolutivos em que se destaca o modelo incremental e o modelo em espiral (Miguel, 2003). Apesar da diversidade existente, vamos apresentar o modelo em cascata e o modelo iterativo que são considerados os modelos base dos restantes processos.

As metodologias utilizadas no desenvolvimento de *software* partilham habitualmente as atividades básicas da arquitetura, como a análise, síntese e avaliação, sendo a maior diferença observada na orientação da metodologia e foco do processo (Hofmeister et al., 2007).

O modelo em cascata (Figura 14) foi a primeira abordagem ao desenvolvimento de *Software*, apresentada em 1970 por Winston Royce (Royce, 1970). Neste modelo as atividades são agrupadas em tarefas e executadas sequencialmente, o que leva a que uma tarefa só tenha sido iniciada após a conclusão da anterior. Este modelo propõe uma abordagem sistemática, linear e sequencial do desenvolvimento de *software*.

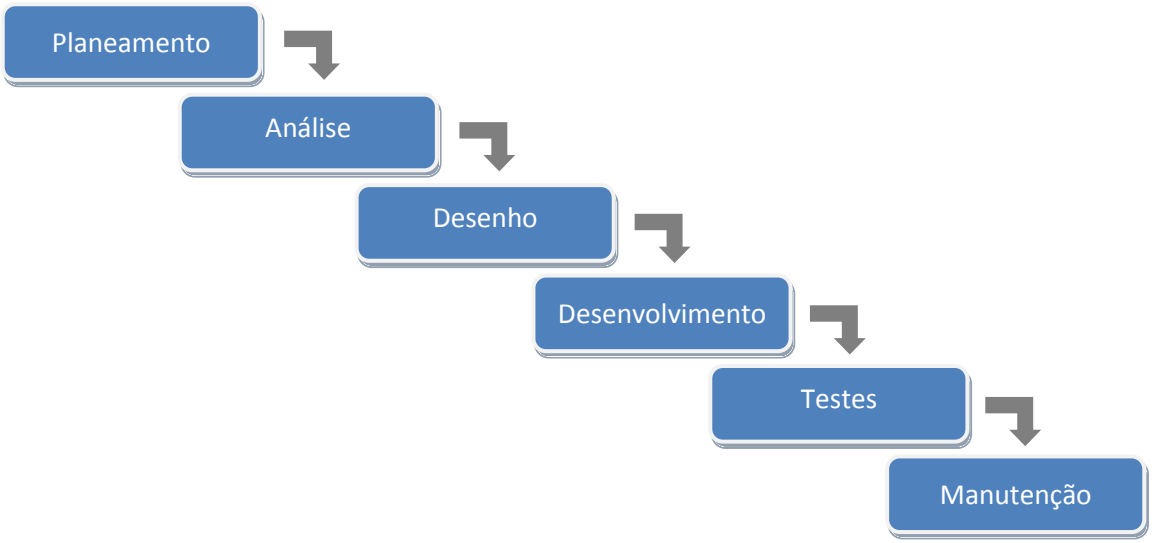


Figura 14 – Processo em cascata (adaptado de Royce, 1970)

No processo de desenvolvimento evolutivo destacámos o modelo iterativo. Os modelos evolutivos possibilitam a construção de versões cada vez mais completas de *software* e os

processos mais recentes de desenvolvimento de *software* são baseados, na sua generalidade, no modelo iterativo. Este modelo promove a comunicação entre todos os intervenientes com o objetivo de promover sistemas finais mais robustos e de qualidade superior. O processo iterativo e incremental permite que a equipa de trabalho possa refinar e alargar pouco-a-pouco a qualidade, o detalhe e o âmbito do sistema envolvido (Silva & Videira, 2001). Neste modelo as sequências lineares estão desfasadas no tempo e cada uma delas produz um incremento utilizável de *software*. São realizadas várias entregas ao cliente, sendo o produto entregue semelhante ao anterior mas com mais algumas funcionalidades implementadas. Este processo é repetido até o projeto estar concluído.

A mais-valia deste modelo consiste em ser implementada uma arquitetura para o *software*, desde o início, que permita agregar todas as funcionalidades desejadas para o sistema, não necessitando de uma reengenharia maciça do sistema de arquitetura (Miguel, 2003).

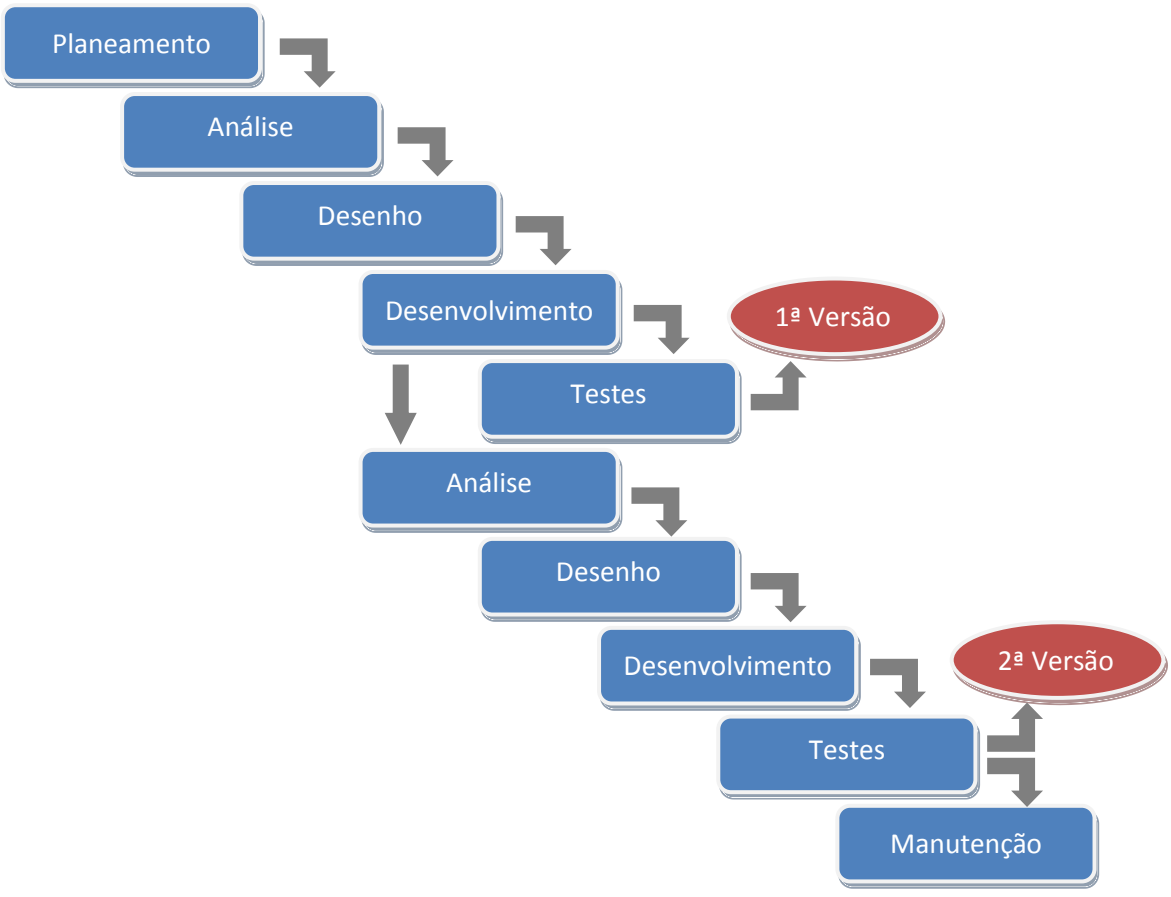


Figura 15 – Processo iterativo, (adaptado de Silva & Videira, 2001)

Em ambos os processos apresentados, na análise é definido o que é requerido para o novo sistema, no desenho define como construir o novo sistema de forma a satisfazer os requisitos e no desenvolvimento é construído o novo sistema tendo em conta a informação do desenho. A fase de testes tem como objetivo verificar se as necessidades dos utilizadores estão satisfeitas e quais são aquelas funções que possuem falhas ou erros. Na última fase, a manutenção, são realizadas mudanças do sistema para corrigir problemas ou atender às necessidades de mudança (Gallegos, et al., 2004).

Após terem sido descritos os vários processos de desenvolvimento de um projeto, vamos definir metodologias ágeis e descrever várias metodologias existentes.

2.2.1. Metodologias ágeis

Um método de desenvolvimento de *software* é denominado método de desenvolvimento ágil de *software* quando é um método em que as pessoas estão concentradas, tem comunicações orientadas, é flexível (pronto para se adaptar à mudança esperada ou inesperada a qualquer momento), rápido (incentiva o rápido desenvolvimento iterativo do produto em versões pequenas), magro (foca-se na diminuição de prazos e custos e na melhoria da qualidade), proactivo (reage apropriadamente às mudanças esperadas e inesperadas), e de aprendizagem (centra-se na melhoria durante e após o desenvolvimento do produto) (Qumer & Henderson-Sellers, 2008a, 2008b). Este método deve ser leve mas suficiente (Cockburn, 2001), ou seja, para cada projeto deve-se encontrar um equilíbrio entre a documentação e registo de informação necessária para o desenvolvimento do projeto e o tempo e custo necessário para o realizar sendo que o registo detalhado de todos os requisitos não é leve e a transmissão exclusivamente oral de conhecimento não é suficiente.

Embora os métodos ágeis estejam em uso na indústria, pouca pesquisa foi empreendida relativamente ao que se entende por agilidade e como um método supostamente ágil pode ser avaliados quanto à sua veracidade a pertencer a esta categoria de abordagens metodológicas de desenvolvimento de *software* (Qumer & Henderson-Sellers, 2008a).

Os princípios de ouro definidos em *agile aliance* em 2001 relativo ao desenvolvimento ágil são (Agile Alliance, 2001; Qureshi & Hussain, 2008):

1. A maior prioridade é satisfazer o cliente através da entrega antecipada e contínua de *software* valioso.
2. As mudanças de requisitos são sempre bem-vindas, mesmo no final do desenvolvimento.
3. Entregar *software* a funcionar, frequentemente num par de semanas ou num par de meses, com uma preferência para o prazo mais curto.
4. O pessoal do cliente e a equipa de desenvolvimento devem trabalhar juntos diariamente durante o projeto.

5. Construir projetos em torno de indivíduos motivados. Proporcionar o ambiente e apoio de que necessitam, e confiar neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informação de uma equipa de desenvolvimento e para o interior da mesma é a conversa cara a cara.
7. *Software* a trabalhar é a principal medida de progresso.
8. Processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, equipa de desenvolvimento e clientes devem ser capazes de manter um ritmo constante indefinidamente.
9. Atenção contínua à excelência da técnica e ao bom *design* aumenta a agilidade.
10. Simplicidade é essencial.
11. As melhores arquiteturas, requisitos e projetos emergem da auto-organização das equipas.
12. Em intervalos regulares, a equipa reflete sobre como se tornar mais eficaz, em seguida, sintoniza e ajusta seu comportamento em conformidade.

Apesar de todos os princípios enumerados, Hanssen & Fægri (2008) que estudaram a produção de *software* através de uma metodologia ágil, consideram que o equilíbrio entre a disciplina e a agilidade é muito importante para a obtenção de bons resultados. Por outro lado, Misra et al. (2009) consideram que os fatores mais importantes para o sucesso de uma metodologia ágil são: satisfação do cliente, colaboração do cliente, compromisso com o cliente, tempo de decisão, cultura corporativa, o controlo, características pessoais, cultura social, de formação e de aprendizagem.

Outros autores encaram a qualidade do *software* de fulcral importância no desenvolvimento de *software*, especialmente quando a sua vida depende todos os dias da qualidade dos serviços produzidos pelos sistemas e dispositivos incorporados ao seu redor (Ovaska, Evesti, Henttonen, Palviainen, & Aho, 2010).

Nesta visão e para a melhoria da qualidade é importante aproveitar as melhorias de processo de *software* fornecidas pelos vários elementos implicadas no desenvolvimento de *software*.

Como referido no ponto 4 dos princípios de ouro de uma metodologia ágil, *software* a funcionar é a principal medida de progresso. Há uma necessidade de colecionar, medir e retirar informações sobre o andamento presente em todos os projetos, e projetos ágeis não são exceção. O processo de controlo é aplicado para assegurar que as expectativas ou planos foram realmente atingidos e é dirigido ao desenvolvimento, aos custos e ao tempo. Os dois principais métodos de controlo são relativos à regulação da alteração da atividade com a conservação dos recursos humanos e físicos da empresa e ativos financeiros. Para esta finalidade costumam-se realizar gráficos ou diagramas de fluxo cumulativos, como por exemplo um gráfico de despesas acumuladas ao longo das várias fases de criação e desenvolvimento de um projeto de *software* com o orçamento planeado e os recursos financeiros realmente utilizados (Gráfico 1) (Agile Alliance, 2001; Meredith & Mantel, 2010).

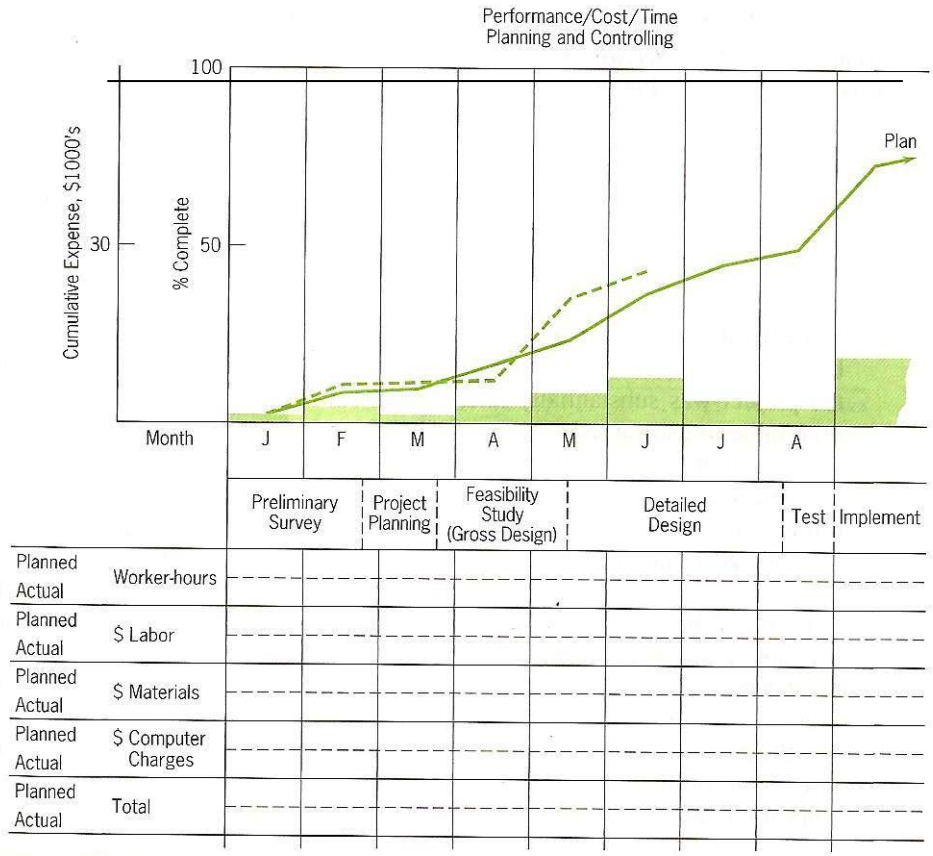


Gráfico 1 – Gráfico de performance integrada custo/tempo (Meredith & Mantel, 2010)

Existem autores que consideraram que estes indicadores não eram claros e propuseram a substituição dos métodos de controlo originais baseados em cálculos de pontos estratégicos com informações dinâmicas extraídas de um sistema de controlo e introduziriam o conceito do plano ideal para medir o progresso em relação a ambos, o final das etapas de iteração e a data de conclusão do projeto (Miranda & Bourque, 2010).

No mundo de metodologias de desenvolvimento de *software*, nenhuma metodologia específica se destaca como dominante e não existe um vencedor claro no horizonte. Segundo Riehle (2001), para sobreviver, prosperar, e crescer, uma metodologia deve ser capaz de aprender com outras metodologias e adaptar com sucesso a novas exigências do mercado.

Existem diversos métodos de desenvolvimento ágil como, por exemplo, *Crystal*, *Dynamic Software Development Method* (DSDM), *Feature-driven*, *Lean software development*, *Extreme programming* (XP; XP2) e *Scrum* (Highsmith & Consortium, 2002; Qumer & Henderson-Sellers, 2008a; Silva & Videira, 2001).

Método *Crystal*

O método *Crystal* foi desenvolvido por Alistair Cockburn e abrange uma coleção de metodologias denominadas pela cor, existindo diversas como, por exemplo, laranja claro, amarelo, vermelho, magenta, azul e violeta, caraterizado pelo tamanho da equipa.

A filosofia deste modelo consiste no desenvolvimento de *software* encarado como um jogo cooperativo de invenção e comunicação, com o objetivo principal de fornecimento de *software*, de trabalho útil, e o objetivo secundário de criação para o próximo jogo.

Duas consequências dessa filosofia é que diferentes projetos precisam ser executados de forma diferente, e a quantidade de modelação e de comunicação que as pessoas precisam fazer é apenas a quantidade necessária para mover o jogo para a frente. Dois valores intrínsecos destas metodologias *Crystal* são: a comunicação centrada e a alta tolerância (Cockburn, 2001).

Método *Dynamic Systems Development Methodology* (DSDM)

O método *Dynamic Systems Development Methodology* (DSDM) foi desenvolvido em meados de 1990, no Reino Unido. O processo de desenvolvimento segundo este método faz-se através de três fases principais: modelo de iteração funcional, desenho e construção iterativa e implementação iterativa. Na fase de modelo de iteração funcional é feita uma recolha e prototipagem de requisitos funcionais com base numa lista inicial de requisitos priorizados. Na fase de desenho e construção, o protótipo é refinado de forma a cumprir todos os requisitos (Highsmith & Consortium, 2002).

Este modelo flexível baseia-se em 9 princípios: a participação ativa dos utilizadores é imperativa, sendo este o princípio fundamental, as equipas devem ser capacitadas para tomarem decisões, deve existir um foco nas entregas frequentes, aptidão para o negócio é o critério de entregas aceites, desenvolvimento iterativo e incremental é obrigatório, todas as alterações durante o desenvolvimento deve ser reversíveis, os requisitos constituem uma *baseline* de alto nível, ou seja, permite a alteração de alguns requisitos durante o processo de desenvolvimento, mas não de todos. Existe um conjunto de requisitos de alto nível, os quais constituem a *baseline*, que não podem ser alterados e permanecem imutáveis ao longo de todo o projeto. Para além destes princípios, o teste é integrada em todo o Ciclo de Vida e a abordagem é colaborativa e cooperativa (Voigt, 2004).

Método *Lean Development* (LD)

O método *Lean Development* (LD) teve origem através dos princípios de produção *lean* usados na indústria automóvel do Japão em 1980. Atualmente tem sido utilizado com sucesso na Europa numa série de projetos de telecomunicações (Highsmith & Consortium, 2002).

Os objetivos do LD são a conclusão em um terço do tempo, dentro de um terço do orçamento e com um terço da taxa de defeito que demoraria o projeto a ser realizado por uma metodologia tradicional, constituindo por estes motivos um grande desafio para a gestão. Para

além destes fatores, consideram que a implementação deste método só terá sucesso se for iniciada pelo topo da organização (Highsmith & Consortium, 2002).

Método *Adaptive Software Development* (ASD)

O método *Adaptive Software Development* (ASD) foi desenvolvido na década de 90 por Jim Highsmith e Sam Bayer e é o mais adequado para utilizar em projetos que lidam constantemente com a mudança. Este modelo utiliza a mudança em vez de lutar contra ela , pelo que necessita de equipas ágeis e facilmente adaptáveis a ambientes de mudança. As práticas da ASD são movidos por uma crença na adaptação contínua em que possuem uma diferente filosofia e um ciclo de vida diferente, orientada para aceitar a mudança contínua como a norma (Highsmith & Consortium, 2002).

Método *Extreme Programming*

O método *Extreme Programming*, também designado por XP pela maioria dos aficionados, foi desenvolvido por Kent Beck, Ward Cunningham e Ron Jeffries (Silva & Videira, 2001). Sendo uma das abordagens ágeis que suscitou mais interesse, defende valores de comunidade, simplicidade, *feedback* e coragem e é definido, pelo menos em parte, pelas suas práticas, nomeadamente, o jogo do planeamento, pequenas entregas, simples *design*, primeiros testes de desenvolvimento, revisão por pares, propriedade coletiva, integração contínua, entre outros.

O interesse pelo XP costuma manifestar-se nos programadores e *testers* que, cansados dos processos demorados, onerosos e formais, optam por uma metodologia que permite entregar *software* de alta qualidade mais rapidamente e de forma mais flexível (Highsmith & Consortium, 2002). Este método enfatiza a comunicação e coordenação entre os membros da equipa em todos os momentos, e requer a cooperação entre a equipa de gestão de clientes e desenvolvimento para formar a cultura empresarial favorável para a implementação bem-sucedida do XP (Qumer & Henderson-Sellers, 2008a).

A chave da filosofia XP é que o Planeamento não é um evento único, mas um processo constante de reavaliação e correção de curso em todo o ciclo de vida do projeto (Beck & Fowler, 2000).

Método *Scrum*

O método *Scrum* é um método que se concentra na gestão de projetos em situações onde é difícil planear com antecedência, com mecanismos de ''controlo de processos empíricos '', onde o *feedback* de *loops* em sequência constitui o elemento central. O *software* é desenvolvido por uma equipa de auto-organização em incrementos (chamados de "*sprints*''), começando com o planeamento e terminando com uma *review*. Os recursos a serem implementadas no sistema estão registados numa lista *backlog*, numa carteira. Então, o *product owner* decide quais itens do

backlog devem ser desenvolvido na *sprint* seguinte (Figura 16). Os membros da equipa devem coordenar o seu trabalho numa reunião de *stand-up* diário. Um membro da equipa, o *scrum master*, é responsável pela resolução de problemas que impedem a equipa de trabalhar de forma eficaz (Dybå & Dingsøyr, 2008).

O planeamento de uma *sprint*, *sprint planning meeting*, é realizado no início de uma *sprint* com a presença da equipa de desenvolvimento, do *scrum master* e do *product owner*. Nesta reunião, o *product owner* estabelece a prioridade dos objetivos, são esclarecidas questões relativas às *user stories* a implementar e descritos mais pormenores relativos às tarefas que constituem a *sprint backlog* (Mountain Goat Software, n. d.-d).

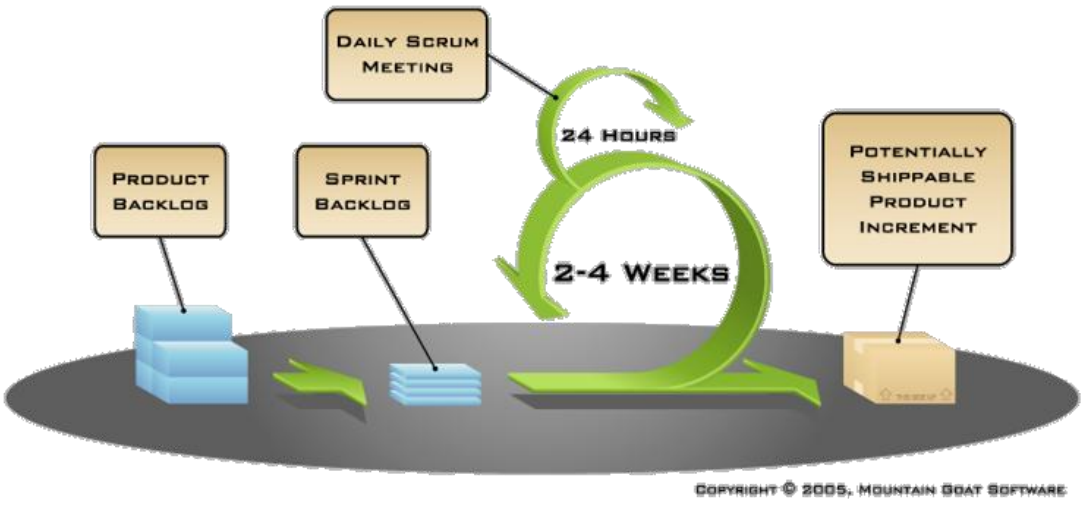


Figura 16 – Scrum (Mountain Goat Software, n. d.-b)

A reunião de *stand-up* diário pode ser denominada "*daily scrum*" e deve ser realizada idealmente, diariamente, no mesmo local, à mesma hora com duração de 15 minutos. Durante a *daily scrum* cada membro da equipa responde às três seguintes questões:

1. O que fez ontem?
2. O que vai fazer hoje?
3. Existem impedimentos para o seu trabalho?

Se surgirem impedimentos, a sua resolução fica a cargo do *scrum master* que os deve resolver o mais rapidamente possível ou atribuir a sua resolução a um elemento que os possa resolver quando o *scrum master* não os poder resolver diretamente (Mountain Goat Software, n. d.-a).

O *product owner*, para além de ter de decidir quais os itens do *backlog* que devem ser desenvolvidos na *sprint* seguinte, tem outras funções. Nessas funções encontra-se a

responsabilidade de criar e manter o *backlog*, estabelecer a prioridade do *backlog* para a próxima *sprint* de acordo com o valor do negócio, ajudar na elaboração das *user stories* sendo a sua responsabilidade estarem preparadas no início da *sprint*. O *product owner* deve transmitir a visão e objetivos no início de cada *sprint*, para guiar a equipa de trabalho no caminho certo e relembrar as metas a alcançar. Por outro lado, deve representar os clientes e envolvê-los no desenvolvimento do produto para garantir que a equipa está a construir o produto que vá de encontro aos objetivos. Deve, também, participar nos *scrums* diários, reuniões de planeamento do *sprint* e, crítica e retrospectiva da *sprint*, sendo uma oportunidade de inspecionar o produto e adaptar a estratégia de desenvolvimento da aplicação. No final de cada *sprint*, o progresso do produto deve ser analisado e o *product owner* tem total autoridade para aceitar ou rejeitar o trabalho feito e também pode alterar a direção e limite de cada *sprint*. O *product owner* é a voz da equipa para o mundo exterior, sendo o incumbido da comunicação para o exterior (Milunsky, 2009).

No final de cada *sprint* deve ser realizada uma reunião de revisão de *sprint* denominada "*sprint review*". Nessa reunião é mostrado o que a equipa desenvolveu, ou seja, as novas funcionalidades implementadas. Esta reunião deve ser informal, não ultrapassar as duas horas de duração e pode ter a presença do *product owner*, *scrum master*, equipa de desenvolvimento, gestores e clientes (Mountain Goat Software, n. d.-d).

Para além destas reuniões, algumas equipas realizam a retrospectiva da *sprint*, "*sprint retrospective*". Esta reunião realizada no final da *sprint*, com a participação de toda a equipa de desenvolvimento, do *scrum master* e do *product owner*, tem como objetivo perceber de que forma é que as equipas podem melhorar o seu desempenho. Existem várias formas de conduzir uma reunião destas, no entanto, Mike Cohn apresenta uma sugestão simples e eficaz: questionar cada elemento da equipa o que se deve começar a fazer, o que se deve parar e o que se deve continuar a realizar. Após estas opiniões, análise e ponderação deve-se realizar uma lista de ações a realizar de forma diferente na próxima *sprint*, para que possa existir uma melhoria do trabalho de toda a equipa (Mountain Goat Software, n. d.-c).

Após a apresentação destas várias metodologias ágeis, descrevem-se várias caraterísticas específicas e particularidades de pequenas empresas, categoria onde se integra a empresa onde foi realizado o estágio, *iUZ Technologies*.

2.2.1.1. Metodologias ágeis em pequenas empresas

Em 1983, Churchill e Lewis (1983) consideraram que as pequenas empresas possuíam 5 fases de crescimento:

1. Existência
2. Sobrevivência
3. Sucesso

4. Descolagem
5. Maturidade de recursos.

Existem poucas pequenas empresas que utilizam modelos que orientam a gestão e implantação das suas iniciativas de melhoria. Isso porque muitos dos modelos de desenvolvimento de *software* não consideram as características especiais das pequenas empresas, como as diferentes capacidades de crescimento, independência de ações, estruturas organizacionais distintas e estilo de gestão distintos (Churchill & Lewis, 1983). Para além destes aspetos, geralmente não são tidas em conta as estratégias apropriadas para a implantação de uma iniciativa de melhorias, nem apresentam um processo explícito para organizar e orientar o trabalho interno dos funcionários envolvidos na implementação das oportunidades de melhoria (Pino, Pedreira, García, Luaces, & Piattini, 2010). Para colmatar esta falha, Pino et al. (Pino, et al., 2010) propuseram uma processo leve para incorporar melhorias que usa a filosofia do metodologia ágil *Scrum*, com o objetivo de dar orientações pormenorizadas de apoio a gestão e o desempenho da incorporação de oportunidades de melhoria nos processos e sua colocação em prática em pequenas empresas. Estes autores descreveram o ciclo de desenvolvimento de um produto de *software* e a sua especificação de requisitos através do diagrama de atividades ilustrado na Figura 17.

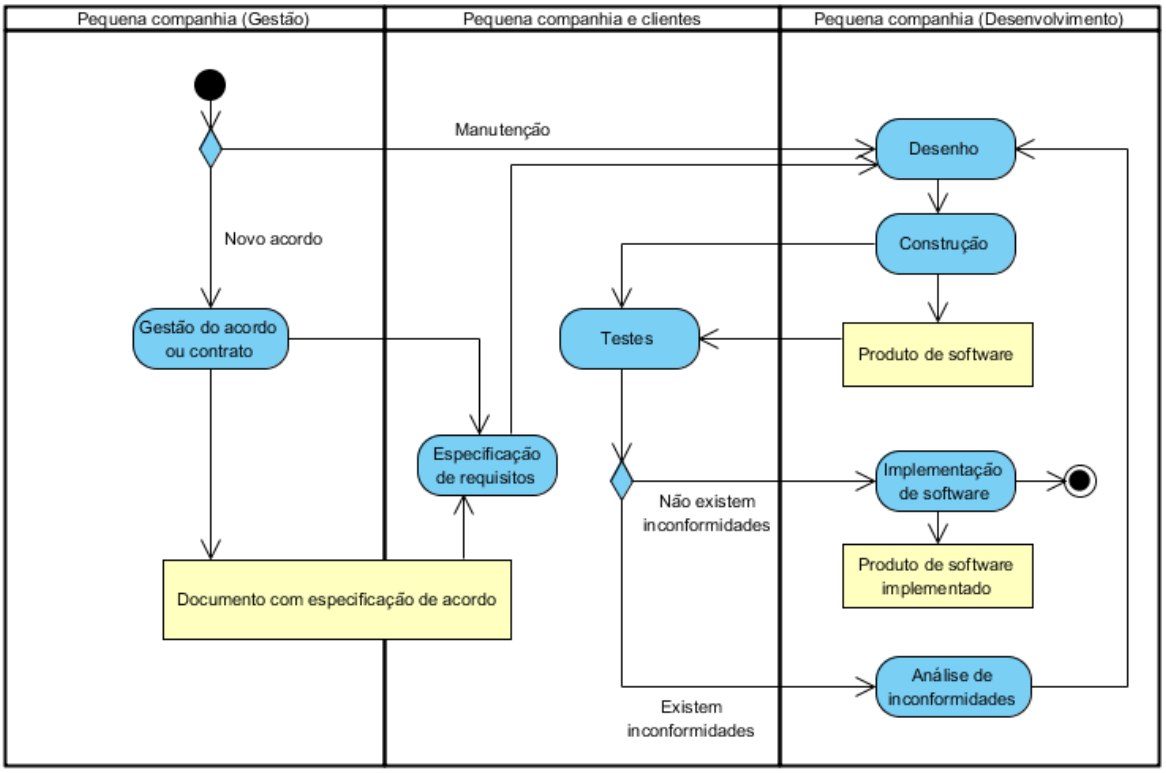


Figura 17 – Exemplo de diagrama de atividade do desenvolvimento de *software* para uma pequena empresa (adaptado de Pino, et al., 2010)

Para uma empresa de *software* cresça e evolua é necessário ter em conta diversos determinantes que são encontrados na literatura. Em seguida serão descritos sucintamente, os determinantes considerados mais importantes na literatura, os quais se encontram agrupados em fatores internos mas também em fatores externos (Nambisan, 2002). Dentro dos fatores internos encontramos:

- Condições na empresa – estratégia tecnológica inicial, recursos financeiros iniciais, entre outros;
- Fatores estratégicos – estratégia do produto, agressividade da estratégia, alianças estratégicas, etc.;
- Recursos e competências – capacidades de gestão, processo de desenvolvimento e ferramentas de marketing, etc. ;
- Caraterísticas internas dos stakeholders – caraterísticas de personalidade, demografia, experiência, orientação para a inovação, etc.

Nos fatores externos deparamos com:

- Caraterísticas industriais – estrutura do mercado, ambiente competitivo, etc.;
- Caraterísticas tecnológicas – tecnologia do ciclo de vida, tecnologia standard, etc.;
- Infraestruturas económicas e tecnológicas – capitais de risco, recursos humanos, infraestruturas de telecomunicação, entre outros;
- Infraestrutura reguladora – incentivos fiscais, regime de propriedade intelectual, etc.;
- Cultura regional e caraterísticas externas dos stakeholders – nomeadamente orientação para inovação, experiência, centros de formação regionais.

As metodologias ágeis desenvolvidas em pequenas empresas devem ter em conta a especificidade e caraterísticas das pequenas empresas e adaptar-se ao meio envolvente.

2.3. Análise e testes

Existem duas fases do processo de desenvolvimento de *software* denominadas Análise e Testes que, tal como foi referenciado nas Figura 13, Figura 14 e Figura 15, serão especificadas neste capítulo.

A fase da Análise inclui a identificação detalhada das funcionalidades do sistema designada por levantamento de requisitos e a respetiva descrição de modo a que os mesmos requisitos possam ser validados pelos utilizadores finais do sistema. Esta fase pode também ser designada de especificação do sistema.

Na fase dos Testes é verificado o sistema no global com o intuito de obter a aceitação do utilizador (Silva & Videira, 2001).

2.3.1. Análise e especificação

A análise e especificação constitui uma fase onde se efetua um estudo detalhado do domínio do problema e termina na elaboração de um documento onde os requisitos funcionais da solução a implementar e outras questões importantes como restrições, âmbito e fluxos de informação são discriminados. Normalmente esta tarefa tem duas sub-tarefas: levantamento de requisitos e especificação do sistema (Silva & Videira, 2001).

Requisitos são uma especificação do que deve ser implementado, constituindo uma descrição de como o sistema se deve comportar, uma propriedade, atributo do sistema, ou podem constituir constrangimentos no processo de desenvolvimento do sistema (Wiegers, 2006). Segundo o Instituto Internacional de Analistas de Negócios (IIBA, n. d.) requisito pode ser definido como:

1. Uma condição ou capacidade necessária para uma das partes interessadas para resolver um problema ou atingir um objetivo.
2. Uma condição ou capacidade que deve ser atendida ou possuída por uma solução ou componente da solução para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos.
3. Uma representação documentada de uma condição ou capacidade como a indicada em (1) ou (2).

Decorre desta definição que os requisitos podem ser implícitos, implicados na definição de outros requisitos, ou estar claramente identificados e definidos (IIBA, n. d.) .

Para realizar a recolha de requisitos existem diversas técnicas como, por exemplo, entrevista com os clientes nas quais realizam as perguntas face a face, a elaboração de questionários, a observação das atividades e do funcionamento do dia-a-dia, a recolha e análise de documentação diversa e a elaboração de pequenos protótipos do sistema que permitam analisar mais facilmente as funcionalidades implementadas (Miguel, 2003).

Existem diferentes tipos de requisitos e diferentes métodos de classificação. Segundo Wiegers (2006), existem requisitos de negócio, requisitos de utilização, requisitos funcionais, requisitos de sistema, regras de negócio, atributos de qualidade, interface externa e restrições.

Os requisitos de negócio representam o “porquê” da informação, porque é que a empresa está a realizar o projeto.

Os requisitos de utilização é um dos tipos que identifica “qual” a informação, nos quais indicam o que é que o utilizador deverá ser autorizado a realizar com a aplicação como, por exemplo, objetivos e tarefas que deverão ser efetuadas. Os requisitos funcionais representam o outro tipo de “qual” informação deve ser implementada, neste caso específico, descrevem que informação a equipa de desenvolvimento deve implementar. Estes requisitos são, por vezes, definidos como requisitos comportamentais e representam o que a aplicação deve fazer ou quais as funções que permite que o utilizador execute.

Para descrever o alto nível de requisitos para um produto que contém múltiplos subsistemas utilizam-se requisitos de sistema. Estes requisitos representam o conjunto de informação que contém subconjuntos de *software* e *hardware* sendo as pessoas uma parte do sistema. Os requisitos de *software* são então representados por requisitos funcionais e não funcionais agregados aos componentes de *software* do sistema.

As regras de negócio incluem as regulamentações governamentais, bem como, as políticas de empresas. Relativamente às características que são valorizadas pelo cliente e pela equipa de desenvolvimento temos os atributos de qualidade, dos quais destacamos a eficácia, execução, usabilidade, eficiência, portabilidade e integridade. Na interface externa entre o sistema e mundo externo são explicitados protocolos utilizados para proteger a informação de outros sistemas. As restrições advêm de opções disponíveis para a equipa de desenvolvimento por razões legítimas (Wiegers, 2006).

Os requisitos podem ser realizados de várias formas, nomeadamente de forma tradicional, através de *use cases* ou de *user stories* (Suscheck, 2012).

Os requisitos tradicionais são geralmente consideradas como capacidades e limitações do sistema. Todos os bons requisitos descrevem o que o sistema pode fazer ou não deve fazer, mas os requisitos que focalizam intensamente no sistema tendem a não enfatizar a interação do utilizador ou do contexto de negócio relacionado com o utilizador ou com a empresa (Suscheck, 2012).

De acordo com o IIBA, os requisitos bem elaborados devem ser completos, testáveis, consistentes, independentes do projeto e inequívocos. Devem ser tão completos quanto possível de forma a não ter lacunas na informação e dar oportunidade para interpretação. Os requisitos devem ser testáveis permitindo a criação de um teste ou uma espécie de prova para garantir que a exigência foi cumprida. Os requisitos devem ser independentes do projeto, ou seja, devem especificar o sistema deve ou não deve fazer, mas não na forma como o *software* irá garantir que a obrigação é cumprida, esta parte pertence ao *design*. Por fim, os requisitos devem ser inequívocos, ou seja não permitir que alguma coisa possa ser interpretada de forma diferente do que pretendia (IIBA, n. d.; Suscheck, 2012).

Para auxiliar a especificação de requisitos, existem autores que recomendam a elaboração de diagramas de transição de estados. Estes diagramas em linguagem UML, orientada para análise de objetos e desenho, permitem que *stakeholders* que não estejam muito familiarizados com o projeto, clientes e autoridades de certificação compreendam as suas diversas funcionalidades, possibilidade de ações e ciclo dos diversos estados (Bouabana-Tebibel & Belmesk, 2007; Idani & Ledru, 2006). Outros autores propõem, para além dos diagramas de estados em linguagem UML, a realização de diagramas de colaboração para enriquecer a compreensão do projeto (Laleau & Polack, 2008). Estes diagramas são também utilizados para auferir se a informação presente nos diagramas é consistente com o realizado no projeto.

A falta de consistência é um problema que têm assolado o desenvolvimento de sistemas de informação desde a sua infância. Lucas et al. (2009) consideraram o modelo UML um dos mais consistentes mas, como os outros modelos, apresenta algumas limitações e falhas. As principais características incluídas na proposta destes autores são: extensibilidade, alinhamento com a proposta do MDA (*Model Driven Architecture*), uma base formal e uma ferramenta *CASE* de integração (Lucas, et al., 2009).

A engenharia de requisitos contém dois componentes importantes: desenvolvimento de requisitos e gestão de requisitos, ilustrado na Figura 18.

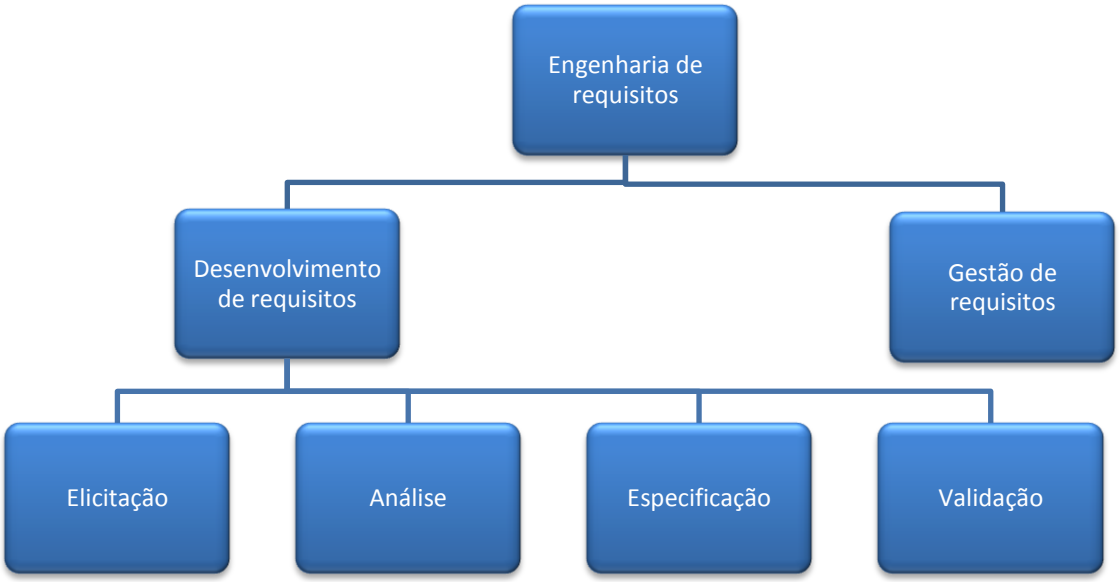


Figura 18 – Subcomponentes de engenharia de *software* (Wiegers, 2006)

O desenvolvimento de requisitos inclui elicitação, análise, especificação e validação. A elicitação consiste em perceber os clientes e entender as suas necessidades (Wiegers, 2006)

A análise consiste na sua maioria no detalhe dos requisitos de alto nível e inclui protótipos, análises gráficas, testes, negociação de prioridades, encontrar requisitos que faltam, avaliar técnicas de fiabilidade e risco (Wiegers, 2006). Segundo o Instituto Internacional de Analistas de Negócios (IIBA, n. d.), a análise de Requisitos descreve como progressivamente elaboram e definem a solução, a fim de permitir que a equipa do projeto desenhe e construa uma solução que irá satisfazer as necessidades do negócio e as partes interessadas. Para fazer isso, temos que analisar os requisitos estabelecidos de nossos parceiros para assegurar que eles estejam corretos, avaliar o estado atual da empresa para identificar e recomendar melhorias, e, finalmente, verificar e validar os resultados.

Na especificação são utilizadas linguagens diversificadas, como, por exemplo, linguagem natural, tabelas e também modelos de análise gráfica.

Finalmente, na validação é garantido que os requisitos estão corretos e todas as necessidades satisfeitas. A validação permite ao analista reescrever alguns requisitos e corrigir ou refinar o documento de requisitos.

A gestão de requisitos inicia quando a equipa acredita que os requisitos são suficientemente bons para servirem de fundação para o desenho e construção de algumas partes do produto (Wiegers, 2006).

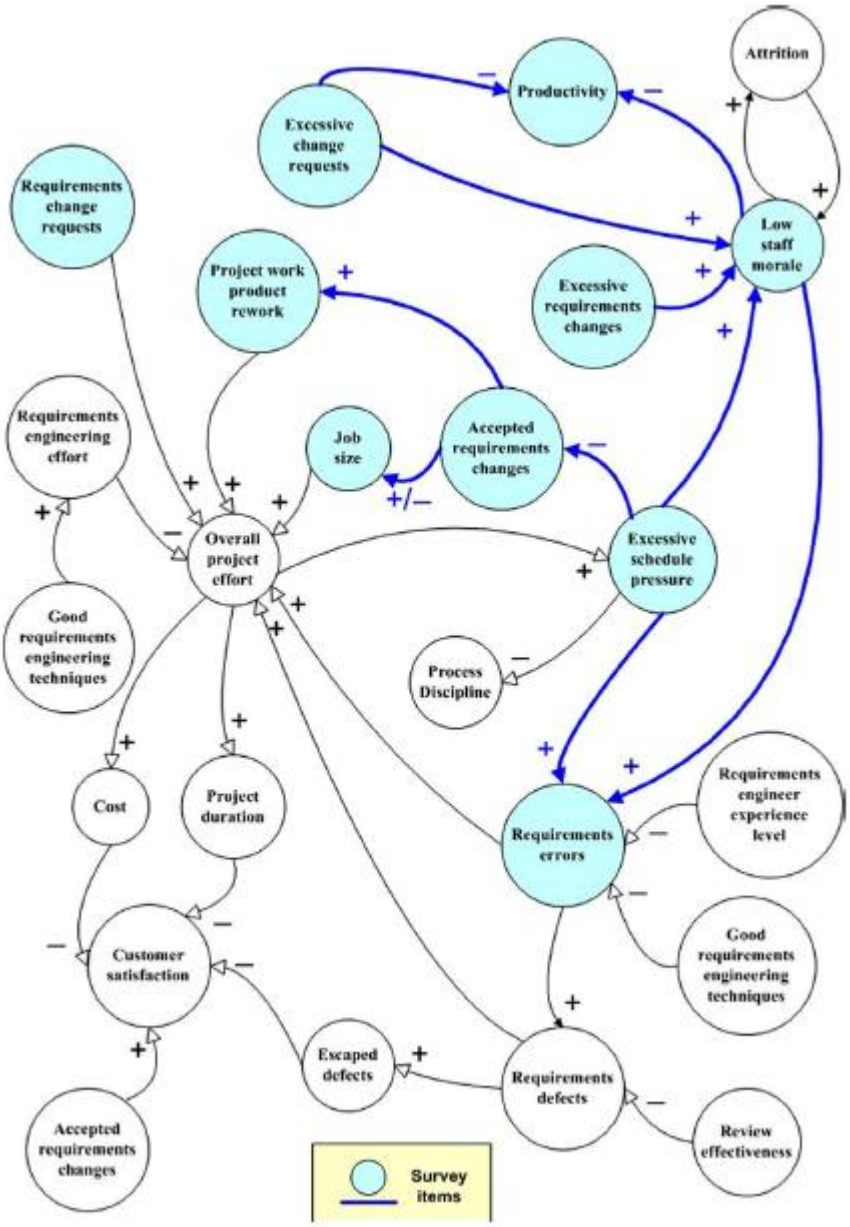


Figura 19 – Modelo de causas de alteração de requisitos (Ferreira, Collofello, Shunk, & Mackulak, 2009)

Habitualmente, durante o desenvolvimento de um projeto realizam-se alterações nos requisitos. Essas alterações e a volatilidade dos mesmos podem ter diversos efeitos na execução do projeto, como, por exemplo, erros e defeitos nos requisitos, maior densidade de defeitos de

requisitos e erros propagados em vários *milestones* (Ferreira, et al., 2009). Apresenta-se na Figura 19 os principais efeitos da alteração de requisitos.

Ao analisar a Figura 19 observamos que existem diversas consequências da alteração de requisitos. Por um lado, a satisfação do cliente aumenta mas, por outro lado, o aumento dos custos, da duração do projeto e erros que não foram detetados, advindos da alteração de requisitos, provocam uma diminuição da satisfação do cliente. Outras consequências detetadas são: erros nos requisitos, excessiva pressão para cumprirem as metas estabelecidas, diminuição moral dos trabalhadores, diminuição da produtividade e um aumento do esforço geral do projeto. Todos estes fatores levam a que tenham de ser muito bem ponderada, qualquer alteração de requisitos.

Neste subseção foi apresentada a definição de requisito e descritos os vários tipos de requisitos. Posteriormente foi apresentada a engenharia de requisitos especificando o desenvolvimento de requisitos e gestão de requisitos. Para além de especificar e gerir os requisitos é necessário realizar testes para verificar se os requisitos estão implementados corretamente, temática que será apresentada em seguida.

2.3.2. Testes e verificação

A fase de testes têm como principal objetivo verificar se as funcionalidades, objetivos e restrições identificados nos requisitos estão a ser contemplados corretamente na aplicação. Existe uma grande diversidade de testes que podem ser atribuídos a diferentes elementos da equipa de desenvolvimento ou mesmo ao cliente.

Nesta seção indicamos uma abordagem estratégica para teste de *software* que é aplicável à maioria dos projetos de desenvolvimento de *software*. O processo recomendado começa por testes unitários, passa para testes de integração, em seguida para testes de sistema e finalmente, testes de aceitação (Pressman, 2000). Na Tabela 3 encontra-se uma pequena descrição dos objetivos dos vários tipos de testes mediante as componentes do sistema que serão alvo de identificação e identifica o perfil que deve realizar esses testes.

Existe uma grande diversidade de modelos de testes. Como foi referido anteriormente, a especificação de requisitos, pode ser realizada de várias formas, nomeadamente, de forma tradicional, por *use cases* ou por *user stories*. Os modelos de testes estão interligados com a forma de realizar a especificação de requisitos. Como existe diversidade na forma de especificar os requisitos, existe uma diversidade ainda maior de realizar os testes. Por exemplo, Weiglhofer, Fraser e Wotawa (2009) propõem o modelo de especificação LOTOS que podia ser utilizado de modo eficiente de forma a ser utilizado em casos de testes e utilizar as informações de cobertura para minimizar os conjuntos de testes enquanto estes são gerados.

Teste	Destinatário	Objetivos
Testes unitários	Programador	• Testar componentes aplicacionais de forma isolada realizados em parcelas do sistema
Testes de integração	Analista Controle de qualidade	• Garantir que os vários componentes que constituem o produto interagem de forma adequada uns com os outros; • Parte fulcral dos testes para a qual se recomenda um plano de testes;
Testes de sistema	Analista Controle de qualidade	• Verificar se foram cumpridos todos os requisitos especificados a nível global, em todos os componentes do produto;
Testes de aceitação	Utilizadores	• Verificar se o produto contém todos os padrões de bom funcionamento estabelecidos no início do projeto.

Tabela 3 – Tipos de teste mediante o âmbito dos componentes que são alvo e verificação (Pressman, 2000; Silva & Videira, 2001)

Apesar desta diversidade, vamos concentrar-nos nos *use cases* e nas *user stories*.

Um *use case* é uma descrição generalizada de um conjunto de interações entre o sistema e um ou mais agentes, onde um agente representa um utilizador ou outro sistema. O nome indicado no *use case* indica o valor que esse *use case* transmite a um ator. Normalmente um *use case* inicia-se com um verbo, contêm um objeto e pode ter, ou não, advérbios e adjetivos Um *use case* pode apresentar muita informação, nomeadamente, atores, descrição, precondições, ciclo normal, ciclo alternativo, exceções, prioridade, frequência de utilização, regras de negócio, requisitos especiais, suposições, notas e erros (Wiegers, 2006).

Na metodologia XP, em 1998, foi introduzida a prática de expressar exigências na forma mais leve, realizando descrições curtas de funcionalidade contadas a partir da perspetiva de um utilizador que são valiosos para qualquer utilizador ou cliente do *software*. Estas descrições foram apelidadas de *user stories*,(Cockburn, 2001; Cohn, 2004).

Existem várias diferenças entre um *use cases* e uma *user story*. Ambos devem indicar qual o seu âmbito mas o âmbito do *use case* geralmente é de maior dimensão que o âmbito de uma *user story*, ou seja, um *use case* tipicamente inclui muito mais informação do que uma *user story*. Uma das mais óbvias diferenças entre as *user stories* e os *uses cases* é o seu alcance. Geralmente um *use case* abrange um propósito muito maior do que uma *user story*, sendo a *user story* de menor porte, não se esperando que ocupe mais de 10 dias de trabalho de desenvolvimento.

Outra diferença é o nível de abrangência, sendo os *use cases* mais completos em detalhes que as *user stories*. Outra importante diferença é a sua longevidade. Para além destas diferenças, os *use cases* destinam-se a viver durante a vida de um produto enquanto as *user stories* são descartadas após o uso. Finalmente, os *use cases* são geralmente escritos como resultado de uma atividade de análise, enquanto as *user stories* são escritas como notas que podem ser usados para iniciar a realização da análise.

As *user stories* enfatizam a comunicação verbal, podem ser utilizadas prontamente no planeamento do projeto permitindo estimativas de dificuldade ou tempo que demora a desenvolver. Os *use cases*, por outro lado, são geralmente demasiado grandes para fornecer estimativas úteis sobre duração do seu desenvolvimento. Para além destas vantagens, as *user stories* permitem que se poupe nos detalhes, ou seja, só é necessário detalhar a *user story* no momento do seu desenvolvimento (Cohn, 2004).

As *user stories* têm como princípios serem entendidas pelos clientes e não representar mais do que um acordo entre clientes e equipa de desenvolvimento alcançado pela discussão e diálogo. Cada *user story* deve acrescentar valor para o cliente e possuir um tamanho que permite realizar várias numa iteração. As *user stories* deveriam ser independentes uma das outras, mas como este facto não é possível, deve-se proporcionar a maior independência possível permitindo uma maior flexibilidade na ordem do seu desenvolvimento. Cada *user story* deve ser testável, possibilitando considerar se a *user story* está implementada ou não (Beck & Fowler, 2000).

Comparando as *user stories* com os requisitos tradicionais também existem algumas diferenças, nomeadamente, no facto de nos requisitos tradicionais faz-se o levantamento exaustivo de todos os requisitos, o cliente aprova e depois inicia-se a implementação do projeto. Esta abordagem faz com que se utilize muito tempo na discriminação de todos os requisitos e que seja difícil estimar o preço de uma funcionalidade do projeto (Cohn, 2004).

Como o progresso de um projeto é demonstrado pela entrega de código testado, as *user stories* como são testáveis, valiosas para o cliente e suficientemente pequenas para que possam ser desenvolvidas várias em cada interação, são as eleitas por alguns autores para a realização dos testes (Beck & Fowler, 2000; Cohn, 2004).

Beck e Fowler (2000) descrevem que imprimem as *user stories* em cartões e as colocam num quadro com os vários estados, permitindo a sua manipulação pela equipa de desenvolvimento durante as sessões de planeamento.

Independentemente da forma de realizar os testes, irão sempre surgir erros. A complexidade do *software* e programas de desenvolvimento acelerados torna difícil evitar os erros. Boehm e Basili (2001) referem 10 regras que podem ajudar a reduzir as falhas no código e que serão apresentadas em seguida:

1. Localizar e corrigir um problema de *software* após a entrega do projeto é 100 vezes mais caro do que encontrar e corrigi-lo durante a fase de requisitos e desenho do projeto.
2. Projetos de *software* atuais gastam cerca de 40 a 50 por cento de seu esforço em refazer trabalho evitável.
3. Cerca de 80 por cento de refazer trabalho evitável vem de 20 por cento dos defeitos.
4. Cerca de 80 por cento dos defeitos vêm de 20 por cento dos módulos e, cerca de metade dos módulos são isentos de defeitos.
5. Cerca de 90 por cento do tempo parado resultam de, no máximo, 10 por cento dos defeitos.

6. As análises por pares detetam 60 por cento dos defeitos.
7. Análises com foco nos objetivos apanham 35 por cento mais defeitos do que análises não direcionadas.
8. Práticas pessoais disciplinadas podem reduzir as taxas de introdução de defeitos até 75 por cento.
9. Desenvolver produtos de alta confiabilidade de *software* custa 50 por cento mais na instrução do que para desenvolver produtos de baixa confiabilidade de *software*. No entanto, o investimento vale mesmo a pena se o projeto envolver operações e custos significativos de manutenção.
10. Cerca de 40 a 50 por cento dos programas do utilizador contêm defeitos não triviais.

Um esforço significativo tem sido colocado no desenvolvimento da educação formal de *frameworks* e ferramentas práticas para a geração de casos de teste. Embora estas técnicas funcionem muito bem em relação à velocidade de teste, o facto dos propósitos dos teste terem de ser manualmente formulados fazem com que o processo de teste dependa da capacidade do testador (Weiglhofer, et al., 2009).

Ao longo da temática de testes foram descritos vários tipos de teste mediante o âmbito dos componentes que são alvo e verificação. Em seguida foram definidos *use cases* e *user stories* e comparados com os requisitos funcionais. Finalmente foram apresentadas algumas medidas para reduzir as falhas de código.

2.4. Conclusões

O enquadramento teórico apresentado centra-se em 3 grandes temáticas: organização em equipas e em projeto, modelos de desenvolvimento de *software* e, análise e testes.

Na organização em equipas e em projetos foi definido o ciclo de vida de um projeto do âmbito geral, as diferentes fases de uma gestão de projetos e as várias arquiteturas organizacionais possíveis.

Este enquadramento inicia-se no âmbito geral, sendo os vários conceitos apresentados para um projeto geral. No ciclo de vida do projeto apresentam-se dois ciclos diferentes transmitindo a diversidade que se pode encontrar e apresenta-se a variação de custos e o nível de pessoal necessário ao longo de todo o ciclo de vida de projeto.

Na gestão de projetos foram apresentados as várias fases e tarefas que se costumam realizar. Posteriormente foram expostos os vários modelos de arquitetura organizacional, especificando as diversas caraterísticas de cada um dos modelos, apresentando, nomeadamente, a arquitetura por projeto que é considerada a ideal para empresas que estão organizadas por projetos de grande dimensão, únicos e diferenciados.

Em seguida, o enquadramento teórico apresentado é específico para projetos de desenvolvimento de *software*. Este temática de gestão de projetos de desenvolvimento de *software* inicia com a descrição das diversas fases e tarefas para o desenvolvimento de *software*.

Em seguida descreveram-se dois modelos que serviram de base a muitas metodologias de desenvolvimento de *software*: um modelo tradicional denominado modelo de cascata e um modelo evolutivo apelidado de modelo iterativo. Após esta breve introdução, apresentou-se a definição e caraterísticas dos modelos ágeis e diversas metodologias ágeis, nomeadamente a *scrum*. Para finalizar esta temática apresentaram-se várias caraterísticas aplicáveis a pequenas empresas, como é o caso da *iUZ Technologies*.

Na última temática abordada, descrevem-se as fases de análise e testes do desenvolvimento de *software*. Na parte de análise e especificação, definem-se os requisitos e descrevem-se vários tipos existentes. Descreve-se a engenharia de requisitos e várias consequências da alteração de requisitos. Na parte de testes e verificação, descrevem-se várias formas para verificar se os requisitos, descritos e implementados na fase anterior, estão a funcionar corretamente no projeto. Para isso, descrevem-se vários métodos possíveis de verificação, no entanto, o modelo baseado *user stories* é o que apresenta maiores vantagens e consegue melhores resultados.

3. Metodologias de trabalho

Ao longo deste capítulo, apresenta-se o planeamento inicial do estágio curricular realizado na *iUZ Technologies* de "Análise, especificação e verificação de implementação de requisitos de *software*". Neste planeamento descrevem-se os objetivos do estágio curricular, especificando as diversas atividades que se pretendiam realizar ao longo do estágio.

Após a descrição detalhada das tarefas é apresentada uma confrontação entre as tarefas planeadas e as realizadas, sendo indicadas vários motivos que suscitaram a alteração do planeamento.

De seguida, apresenta-se o planeamento inicial do estágio curricular.

3.1. Planeamento de estágio

O estágio curricular denominado "Análise, especificação e verificação de implementação de requisitos de *software*", enquadrado nas disciplinas relacionadas com o Planeamento Estratégico de Sistemas de Informação foi realizado na empresa *iUZ Technologies*, sendo que teve a duração de 8 meses, com o seu início em 26 de Setembro de 2011 e o seu término em 25 de Maio de 2012.

Este estágio teve orientação do Mestre Licínio Kustra Mano na empresa *iUZ Technologies* ficando a orientação pedagógica a cargo do Professor Doutor Daniel Polónia.

Os objetivos que se inicialmente se pretendiam alcançar com o estágio curricular são apresentados, em seguida, de acordo com o plano de trabalho detalhado oportunamente proposto pela empresa. É igualmente apresentado o mapa de Gantt que suportou a planificação temporal do estágio.

Em termos gerais o plano de trabalho dividiu-se em seis grandes áreas que passamos a detalhar:

A.1) [5s: 1-5] Aprofundar conhecimentos sobre a atividade económica da empresa.

- Estudar *site* da empresa;
- Conhecer soluções em carteira;
- Conhecer tipologia de projetos realizados;
- Elaborar uma breve análise sobre a visão pessoal da empresa;

A.2) [4s: 3-7] Compreender os procedimentos e metodologias utilizadas nos processos produtivos.

- Estudar e compreender o que é o processo inerente à Engenharia de *Software*;
- Conhecer metodologias de Engenharia de *Software*;
- Perceber as etapas e perfis envolvidos no processo;
- Analisar casos exemplificativos de especificação de requisitos;

- Elaborar especificação de requisitos simplificados (em projeto a definir);
- Elaborar plano de testes de aceitação (em projeto a definir);
- Realizar testes de aceitação (em projeto a definir);

A.3) [18s: 3-21] Participar em atividades de análise e especificação de requisitos de soluções a desenvolver.

- Participar na análise da atividade comercial do cliente;
- Elaborar a especificação detalhada da solução a implementar;
- Elaborar plano de testes de aceitação;
- Realização de testes de aceitação;
- Identificar e recomendar melhorias e otimizações ao processo de testes;

A.4) [14s: 9-23] Realizar tarefas integradas no processo produtivo da empresa:

- Verificar de requisitos, realizar de testes de qualidade e aceitação.
- Rever plano de testes de aceitação, tendo por base a especificação disponibilizada (em projeto a definir);
- Realizar e documentar testes de aceitação (em projeto a definir);

A.5) [6s: 22-29] Elaborar documentação de apoio aos utilizadores.

- Preparar recursos de apoio ao utilizadores, nomeadamente:
    - Manuais de utilizador;
    - Guiões multimédia;

A.6) [7s: 29-36] Elaboração de relatório de estágio

- Escrever relatório;
- Rever relatório;
- Entregar relatório;

No ponto A.5 é apresentado um objetivo de preparar guiões multimédia como recurso de apoio aos utilizadores. Este objetivo é, na realidade, uma sugestão, uma possibilidade de inovação. Por este motivo, não é considerada necessária a sua realização para alcançar os objetivos propostos.

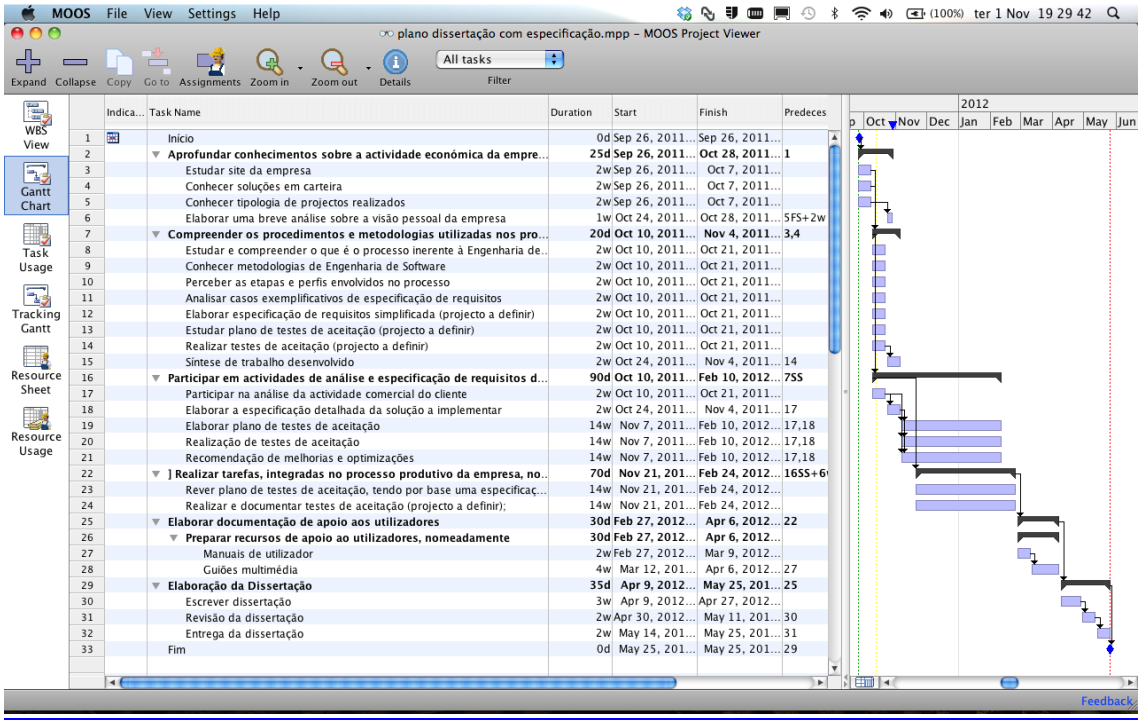


Figura 20 – Mapa de Gantt com a previsão do estágio

3.2. Realização do estágio

Ao longo do estágio verificou-se que o planeamento do estágio não foi cumprido, tendo-se verificado diversas alterações, como se pode observar na Tabela 4.

O estágio curricular foi iniciado com o aprofundamento de conhecimentos sobre a atividade económica da empresa em que estudei o *site* da empresa, conheci soluções que a empresa tinha em carteira e observei vários projetos que tinham sido realizados na empresa. Neste âmbito, observa-se que existe bastante diversidade de projetos, como pode ser consultado na secção 4.1.3 em que se especificam os diversos produtos desenvolvidos na *iUZ Technologies* e na secção 4.1.4 em que se descrevem os vários mercados em que a *iUZ* já trabalhou e apresenta os diversos mercados onde está inserida a sua atividade a uma pequena previsão relativa à evolução da empresa.

Na minha opinião, a *iUZ Technologies* é uma empresa muito jovem, com um espírito muito leve, ágil, com equipas de trabalho muito cooperativas, dinâmicas, inovadora, criativas e que possuem um grande espírito de entreajuda. A principal crítica apresentada ao *site* da empresa é que, na minha opinião, não transmite a essência da *iUZ* de uma empresa dinâmica, inovadora, jovem e criativa.

Esta tarefa e a seguinte foram realizadas num período de tempo mais reduzido que o inicialmente previsto, pois o projeto *Green* principiou na terceira semana do estágio curricular e participei nas reuniões deste projeto desde a abertura da especificação de requisitos.

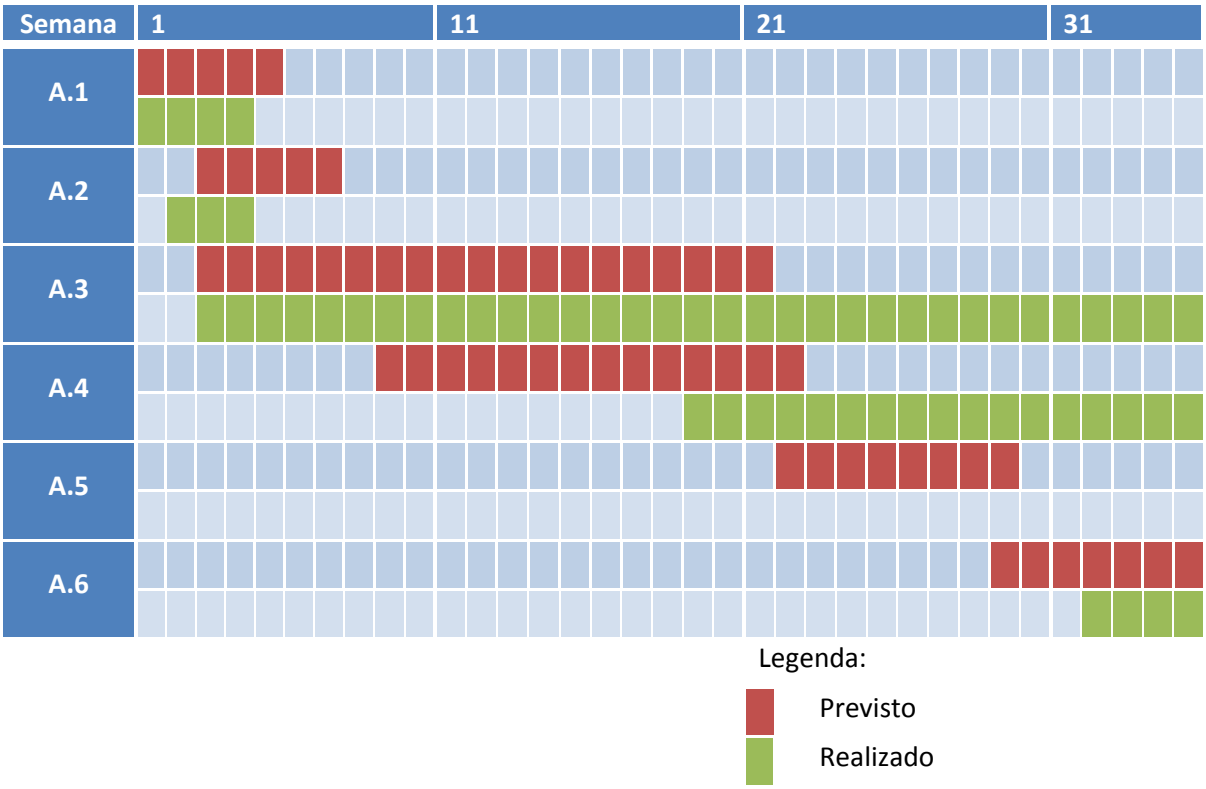


Tabela 4 – Planeamento das atividades previstas e realizadas no estágio curricular

Para o objetivo de compreensão dos procedimentos e metodologias utilizadas nos processos produtivos, realizei a leitura do livro UML, Metodologias e Ferramentas Case (Silva & Videira, 2001) e do livro *More about software requirements: Thorny Issues and Practical Advice* (Wiegers, 2006). Estes livros foram recomendados pelo orientador da empresa porque abordavam as metodologias utilizadas e praticadas na *iUZ Technologies* e continham os processos, metodologias utilizados na engenharia de *software*, bem como, as várias etapas e perfis envolvidos no processo. Realizei testes de aceitação no projeto de Auditoria para experienciar uma nova atividade, a realização de testes de aceitação.

Na terceira semana, dia 10 de Outubro, iniciou a minha participação no projeto *Green*. Apesar de ainda ter continuado com a realização de testes de aceitação no projeto Auditoria por mais uma semana, o meu trabalho realizado passou a concentrar-se no projeto *Green*.

Neste projeto participei em diversas atividades como a análise e especificação de requisitos de soluções a desenvolver participando na análise da atividade comercial do cliente, a especificação detalhada da solução a implementar. Nestas diversas tarefas, realizei *use cases* que acabaram por não ter muita utilização, diagramas UML de estados e de atividade para ilustrar e clarificar determinadas funções. Transformei os requisitos em *user stories* para poderem ser desenvolvidos pela equipa de desenvolvimento. Estas tarefas foram realizadas até ao término do estágio curricular.

Para além das atividades mencionadas relativas à análise e especificação de requisitos, realizei atividades relativas aos testes e verificação. Neste âmbito realizei planos de testes de qualidade e de aceitação e testes de aceitação e qualidade, baseados nas *user stories* utilizadas para o desenvolvimento. No final de cada *sprint* verifiquei se as *user stories* estavam bem implementadas e se todos os requisitos estavam cumpridos.

A atividade de elaboração de documentação de apoio para os utilizadores não foi realizada devido ao atraso do desenvolvimento do projeto, uma vez que, esta tarefa deveria ser realizada no final do desenvolvimento do projeto.

Para uma consulta mais detalhada de todas as atividades realizadas ao longo do estágio curricular poderá ser consultado o Anexo 1.

A última atividade de escrita do relatório de estágio foi elaborada ao longo de todo o estágio, mas com mais ênfase nas últimas quatro semanas de estágio, motivo pelo qual se encontra apenas este tempo na Tabela 4.

Fazendo uma análise geral, observa-se que a maioria dos objetivos propostos foram cumpridos, tendo-se verificado um atraso significativo nas atividades que possuíam uma maior duração.

4. Caso de estudo

Este capítulo é referente, concretamente, ao caso de estudo. A temática fulcral do projeto inicia com uma descrição da empresa *iUZ Technologies* apresentando-se uma caraterização dos produtos e serviços disponibilizados pela empresa. Para além desta informação é apresentada a estrutura organizacional da *iUZ* e apresentada uma descrição do projeto "Green", projeto sobre o qual foi analisada a metodologia aplicada na empresa em questão. Finalmente, chegamos à operacionalização do trabalho em que são descritas e analisadas as formas e metodologias de desenvolvimento deste projeto.

O capítulo termina com a síntese das conclusões do mesmo.

4.1. Contextualização da *iUZ Technologies*

A *iUZ Technologies* (Figura 21) é uma empresa de Engenharia de Software que tem como missão concretizar soluções inovadoras de base tecnológica que gerem valor para o cliente, através da integração de informação e de novos paradigmas de interação com o utilizador, bem como entregar aos seus clientes mobilidade, portabilidade e usabilidade em áreas como a identificação, localização e segurança de recursos, envolvendo o cliente ao longo de todo o processo (2011 iUZ Technologies; UATEC, 2011).

Figura 21 – Logótipo da *iUZ Technologies*

Foi fundada em 2007 por quatro sócios, Isabel Cruz, Jorge Moura, Licínio Mano e Luís Barata da Rocha, que desejavam criar uma organização capaz de dar resposta às crescentes exigências do mercado tecnológico (Costa, 2010).

A *iUZ Technologies* investe na criatividade e inovação das suas aplicações através do recurso às mais recentes tecnologias, bem como, à inteligente reutilização de tecnologias amplamente disseminadas.

O compromisso com o cliente tem por base o profissionalismo do projeto, as competências especializadas dos seus colaboradores e num bom funcionamento como equipa.

A combinação destes valores resulta numa orientação objetiva à criação de valor e mais-valias para o cliente. A *iUZ Technologies* atua em diversos mercados e possui uma gama de produtos extensa que será apresentada sucintamente em seguida.

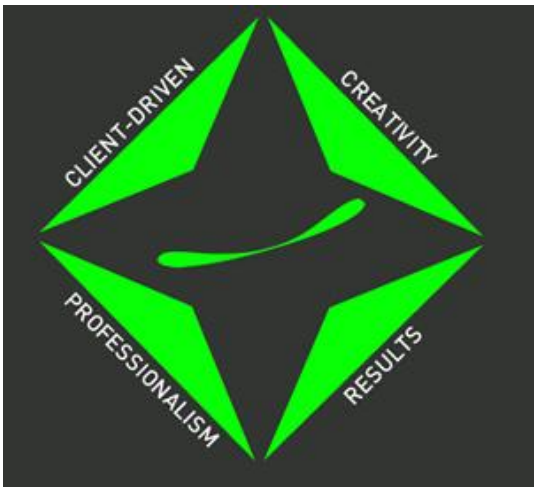


Figura 22 - Valores da *iUZ Technologies* (iUZ Technologies, 2011)

4.1.1. Portfólio

A *iUZ Technologies* possui um campo diversificado de clientes destacando-se na categoria de Organização de Eventos de Desporto, o Sportis, na categoria Governamental, J.F.Luso, na categoria Comercial: Cymbolex e dUOgift e na categoria sem fins lucrativos: AGAP, Propor2008, APGICO e C.A.M.. Estes clientes apresentavam necessidades e objetivos diferentes, os quais serão descritos, em seguida, sucintamente.

O Sportis, em parceria com uma instituição de combate à droga, organizou um evento denominado *Bike Tour Online Experience*, ilustrado na Figura 23 (iUZ Technologies, n. d.-a Bike Tour Online Experience). Este evento promove práticas saudáveis visando mudar hábitos, alertar, definir metas e promover a mudança para um mundo melhor. O projeto teve início em Lisboa e Porto e expandiu para Madrid em 2008 e São Paulo em 2009.

Figura 23 – *Bike Tour Online Experience*

A J. F. Luso, Junta de Freguesia do Luso é uma organização governamental local que pretendia o desenvolvimento de um *site* com informações e serviços para os cidadãos e turistas (iUZ Technologies, n. d.-b J. F. Luso).

Na categoria comercial distinguimos a Cymbolex e o dUOgift.

A Cymbolex (Figura 24) é uma PME que trabalha no âmbito das Energias Renováveis. A *iUZ Technologies* trabalhou com esta empresa com o objetivo de criar a sua presença online e transmitir a mensagem que a empresa queria comunicar (iUZ Technologies, n. d. -c Cymbolex - Isolamentos e Aquecimentos Unipessoal, Lda).

A dUOgift é uma loja de presentes de Portugal conhecida pela variedade, cor e produtos divertidos. A *iUZ Technologies* desenvolveu a presença online da empresa, a qual, com a evolução, evoluiu para um site de eCommerce.

Na categoria sem fins lucrativos surgem vários projetos, dos quais, destacamos a AGAP e o C.A.M..

A AGAP é uma Associação de Empresas de Ginásios e Academias de Portugal que representa mais de 400 empresas distribuídas ao longo de Portugal. Recorreu aos serviços da iUZ para aumentar a sua qualidade, profissionalismo e serviços certificados. (iUZ Technologies, n. d. -a AGAP - Associação de Empresas de Ginásios e

Figura 24 – Cymbolex – Isolamentos e Aquecimentos Unipessoal, Lda.

Academias de Portugal)

O C.A.M., Clube Atlântico da Madalena, é uma organização de Portugal que possui diversas atividades em ação. A *iUZ Technologies* trabalhou com este clube para criar a sua presença online e um sistema de gestão que permitia a realização de tarefas como adicionar equipas e detalhes doa atletas, publicar notícias e eventos diários de desporto (iUZ Technologies, n. d. -b Clube Atlântico da Madalena).

4.1.2. Serviços

Os produtos e serviços disponibilizados pela *iUZ Technologies* resultam do seu posicionamento na cadeia de valor no desenvolvimento e implementação de soluções tecnológicas.

Figura 25 – Posicionamento na Cadeia de Valor

Os principais serviços prestados pela *iUZ Technologies* distinguem-se em:

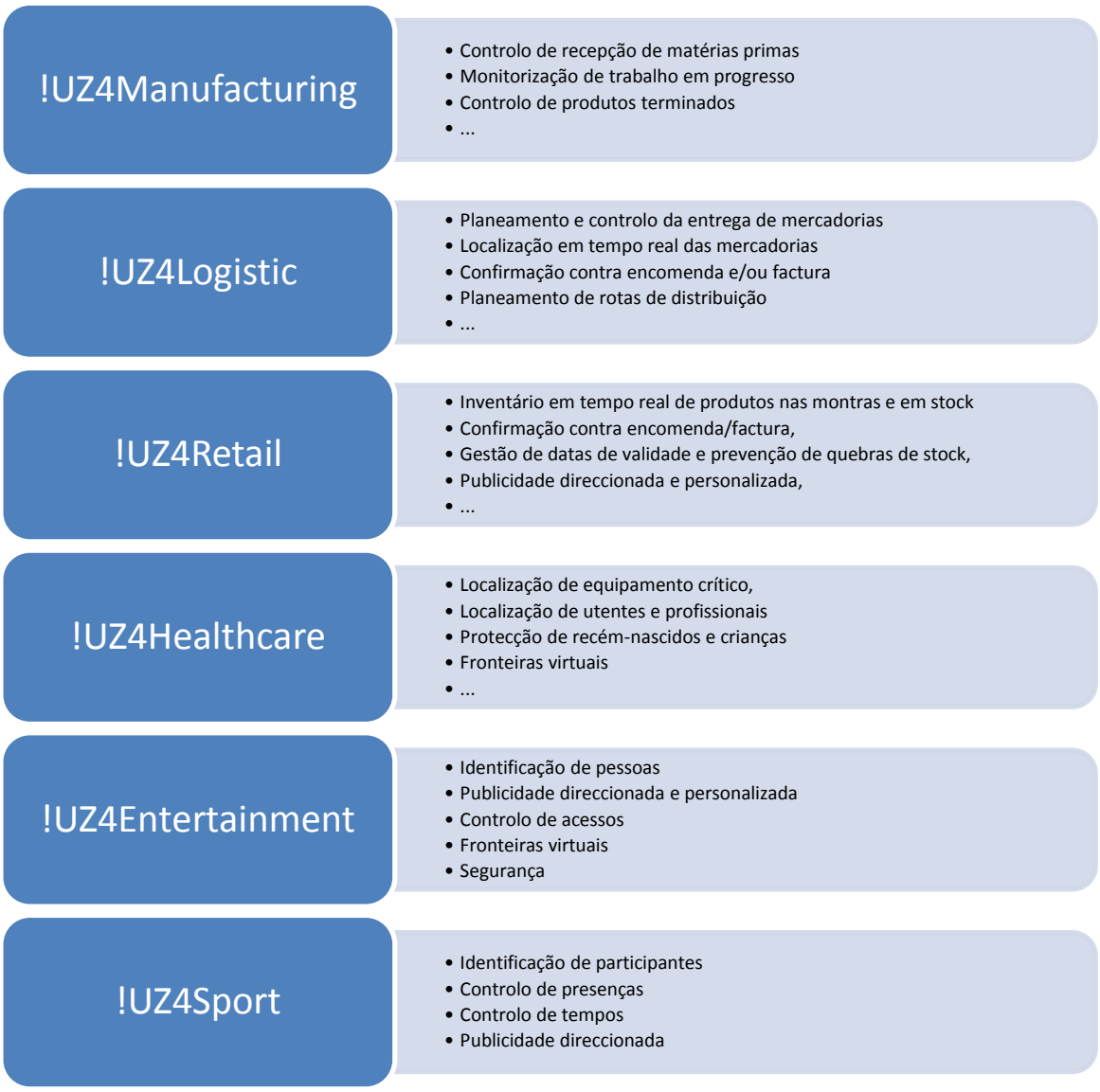
- Desenvolvimento e implementação de soluções para Identificação e Localização de bens e pessoas.
- Consultoria informática;
- Manutenção e suporte de Sistemas de Informação.

A metodologia utilizada pela *iUZ Technologies* será apresentada posteriormente na operacionalização do trabalho. Em seguida serão apresentadas as caraterísticas dos produtos realizados na *iUZ Technologies*.

4.1.3. Produtos

A resolução de problemas específicos de cada atividade de negócio resulta numa gama diversificada de produtos apresentada pela *iUZ Technologies*.

Tendo por base uma plataforma tecnológica comum, cada produto apresenta características específicas adequadas à lógica de negócio do mercado a que se destina.



Quadro 1 – Produtos da *iUZ Technologies*

Cada produto encontra-se em constante evolução no sentido de acompanhar a procura de requisitos e novas funcionalidades por parte dos mercados exigentes e competitivos a que se destinam.

Segue-se a apresentação de vários mercados e uma descrição sumária da evolução da *iUZ Technologies*.

4.1.4. Mercados - Evolução da *iUZ Technologies*

As possibilidades abertas pelas soluções tecnológicas apresentadas pela *iUZ Technologies* permitem a interação com diferentes mercados cujos processos de negócio exigem novos

padrões de modernização. Inicialmente, quando surgiu a *iUZ Technologies*, existiam mercados-alvo, em potencial, muito amplos e diversificados, como mostra a tabela seguinte.

PRODUÇÃO	•Cadeias de produção •Unidades fabris •Industrias têxtil
DISTRIBUIÇÃO	•Centros de armazenamento e/ou distribuição
RETALHO	•Lojas de roupa, •Lojas de CD e/ou DVD de música, •Jogos •Filmes •Perfumarias •Produtos de estética e beleza •Farmácias
SAÚDE	•Hospitais •Centros de saúde •Clínicas •Veterinários •Maternidades
ENTRETENIMENTO	•Publicidade personalizada, •Concertos •Espectáculos •Passeios •Jardins de infância •Centros de terceira idade
DESPORTO	•Atletismo •BTT •Ciclismo, •Desporto escolar, •Provas oficiais ou eventos recreativos

Quadro 2 – Mercados potenciais da *iUZ Technologies*

Ao fim de 5 (cinco) anos de existência, verificou-se uma focalização em três áreas de negócio: Saúde, Transportes e Indústria. Pela observação da diversidade no Portfólio dos produtos e dos clientes, verifica-se que a *iUZ Technologies* utiliza uma estratégia de diferenciação, ao invés de uma estratégia de liderança em custos.

A estratégia de diferenciação consiste num conjunto de ações destinado a produzir ou distribuir produtos/serviços (a um custo aceitável) que os clientes percebam serem diferentes dos da concorrência, e que essas diferenças lhes sejam importantes (Hitt, Ireland, & Hoskisson, 2008) Nesta estratégia é de destacar que, apesar do preço não ser o ponto fulcral da estratégia, este

não pode exceder o que os clientes-alvo estão dispostos a pagar. Os produtos e os serviços prestados são desenvolvidos à medida das necessidades de cada cliente, e os clientes valorizam mais características diferenciadoras do que o baixo custo.

Hitt et al. (2008) consideram que, numa estratégia de diferenciação, é necessário desenvolver novos sistemas e processos, formatar percepções através da publicidade, existir um grande enfoque na qualidade, desenvolvimento de competências em ID e otimização dos inputs humanos através de baixa rotação de pessoal e elevada motivação. Na *iUZ Technologies* verifica-se uma atualização e desenvolvimento de novas técnicas de trabalho e uma elevada preocupação e exigência com a qualidade dos produtos desenvolvidos. A nível de recursos humanos estes possuem uma rotação muito reduzida ou mesmo inexistente e revelam motivação e empenho na realização das suas tarefas. Na minha opinião, o desenvolvimento de competências de ID não são descuradas e existe publicidade. Considero que apenas este ponto da publicidade, poderá ser melhorado e que ainda não está explorado na sua plenitude, nomeadamente no que respeito diz à presença na Internet da empresa que não espelha a atual orientação de negócio assumida pela empresa.

Ao aplicar uma estratégia de diferenciação, a empresa pode acrescentar valor pelas características e atributos únicos do produto, pela elevada qualidade de serviço ao cliente, qualidade superior dos produtos desenvolvidos e pelo prestígio ou exclusividade que proporcionam a inovação rápida.

Existem vários fatores que potenciam a diferenciação. Segundo Hitt, et al (2008) de entre esses fatores destacam-se as características do produto/serviço únicas, as características de desempenho únicas e os serviços excecionais. Na *iUZ Technologies* verificam-se estas características, uma vez que, desenvolvem produtos únicos que vão de encontro às necessidades específicas de cada cliente. Destacam-se ainda a utilização de novas tecnologias, a qualidade dos inputs, e excecionais competências e/ou experiência. Observa-se na iUZ uma procura constante de novas tecnologias, que respondam de melhor forma às necessidades dos clientes e à realização do trabalho interno da empresa, bem como, uma preocupação com a qualidade dos *inputs*. Quanto à experiencia da empresa, esta já exerce atividade há, aproximadamente, cinco anos. Relativamente aos recursos humanos, estes possuem elevadas competências para a realização das suas funções, uma vez que todos possuem uma Licenciatura ou Mestrado, em área científica relevante para o processo de Engenharia de *Software* e, em média, os colaboradores possuem mais de 3 anos de experiência profissional na área de negócio em questão.

Após uma contextualização detalhada da *iUZ Technologies* a nível de portfólio, produtos, serviços e mercados em que atua, passamos a uma descrição da sua estrutura organizacional.

4.2. Estrutura organizacional da *iUZ*

A *iUZ Technologies* está organizada por projetos de grande dimensão. O ciclo de vida dos projetos descreve as seguintes fases:

- Iniciação
- Planeamento
- Desenvolvimento
- Testes
- Correção bugs e Verificação
- Exploração
- Encerramento.

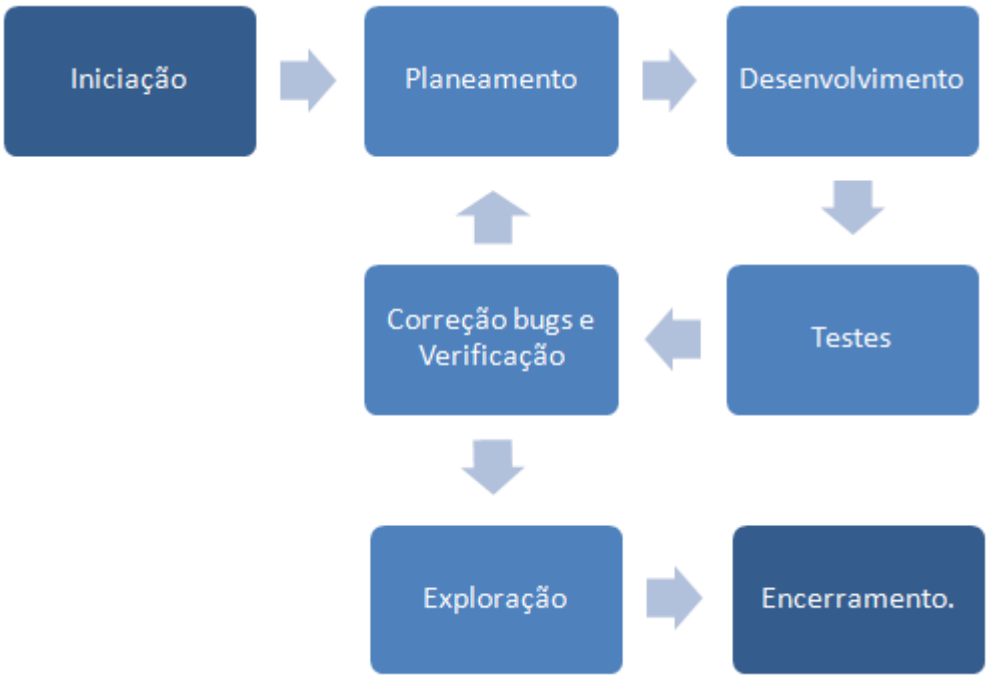


Figura 26 – Ciclo de vida dos projetos da *iUZ Technologies*

No caso particular do projeto apresentado neste documento, verifica-se uma grande importância das fases de planeamento, desenvolvimento e testes. Como o projeto continha um grau elevado de risco, um planeamento muito detalhado e pormenorizado foi o processo implementado para reduzir o mesmo. O facto de o projeto ser inovador também contribuía para aumentar o grau de incerteza do mesmo, a qual também era minimizada pelo planeamento preciso e detalhado.

O desenvolvimento e os testes foram de fulcral importância para verificar se os objetivos pretendidos pelo cliente estavam a ser atingidos e quais as modificações que deveriam ser

implementadas para o projeto realizar todas as tarefas do interesse do cliente e melhorar o seu desempenho e qualidade.

A gestão do projeto descrito inclui:

- Identificação de requisitos;
- Registar as várias necessidades, compromissos e expectativas dos *stakeholders* no planeamento do projeto;
- Realizar o balanço do projeto competitivo registando:
    - Âmbito, Qualidade, Prazos, Orçamento, Recursos e Risco.

Como foi referido anteriormente, o planeamento é uma fase muito importante da elaboração do projeto. Nesta fase realizou-se a identificação e registo dos requisitos do cliente. Após este processo, realizou-se um planeamento detalhado que incluía tarefas, prazos e a especificação das pessoas que iriam realizar a tarefa. Este planeamento estava disponibilizado à equipa de trabalho, num programa de *software*, que poderia indicar se as tarefas estavam cumpridas, em processo, por realizar e, ainda, impedidas de realizar por algum motivo em específico. Estes planeamento permite à equipa de trabalho do projeto fazer uma gestão eficaz para o alcance dos objetivos propostos e cada colaborador identificar corretamente quais as tarefas que tem a desenvolver.

Dada a elevada complexidade e dimensão do projeto foram estabelecidos vários prazos em que a equipa tinha que cumprir uma série de requisitos. Após ser atingido cada um destes pontos, denominados *release*, o projeto era enviado ao cliente com o intuito de ele validar e ou corrigir o trabalho realizado. Este processo permite que o cliente tenha um papel importante na elaboração do projeto e a sua colaboração permita uma atualização constante do projeto.

A arquitetura organizacional da *iUZ* é a organização por projeto. Como já foi referido, a empresa está organizada por projetos de grande dimensão. Estes projetos estão distribuídos por duas equipas dentro da empresa: *Orange Team* e *Green Team*. As equipas têm, praticamente, colaboradores com as mesmas funções, permitido aos mesmos estarem concentrados em projetos específicos, o que permite a melhoria da sua produtividade. Devido à dimensão reduzida da empresa e à especificidade das tarefas, alguns colaboradores pertencem a ambas as equipas. No entanto, pode-se considerar que a arquitetura organizacional é por projeto.

Neste subcapítulo descrevemos o ciclo de vida dos projetos implementado na *iUZ Technologies* e a estrutura organizacional implementada na empresa para o desenvolvimento de projetos de *software*.

Em seguida são descritas as caraterística do projeto que serviu de base para a realização deste relatório.

4.3. Projeto do caso de estudo

A realização deste caso de estudo incidiu sobre a análise de um projeto em particular. Esse projeto será chamando ao longo deste documento como projeto *Green*, nome fictício atribuído devido a questões de confidencialidade e segurança que não permite a exposição detalhada de todos os pormenores do negócio.

O projeto *Green* consistia num projeto diferenciado, criado especificamente para corresponder às necessidades de um cliente. Este projeto consistia na venda de um serviço do interesse de uma grande diversidade etária de clientes. Por este motivo, o projeto desenvolvido teria de ser intuitivo e de fácil utilização para poder ser utilizado por uma elevada variedade de clientes.

O projeto *Green* baseava-se no desenvolvimento de uma aplicação de *software*, cujo objetivo consistia em estar disponível para o público em geral. Esta aplicação era muito completa (em número e tipologia de funcionalidades) e bastante complexa porque reunia numa única aplicação as ofertas existentes, a forma de realizar a procura do serviço almejado e os clientes finais. A aplicação permite que os serviços estejam disponíveis para os clientes finais de uma forma muito acessível, prática, confortável e económica, o que juntamente com a aglomeração de todos os utilizadores descritos anteriormente consistiam nas grandes mais-valias do projeto. Todos os utilizadores: agentes de procura, agentes de ofertas clientes finais e gestores do projeto *Green* teriam acesso a páginas próprias para o seu perfil em que poderiam fazer a configuração, seleção, definição e validação dos seus produtos.

O projeto *Green* tinha inicialmente previsto um desenvolvimento inicial de duração de 20 semanas (Tabela 5), mas devido a atrasos de desenvolvimento e alteração de requisitos, na data de término do estágio curricular, 25 de maio de 2012, ainda estava a decorrer o desenvolvimento do projeto.

ATV	DESCRIÇÃO	INÍCIO	FIM
1	SW Analysis & Specification	2011-10-03	2012-01-27
2	SW Architecture & Design	2011-10-10	2012-01-27
3	SW Implementation	2011-10-24	2012-02-24
4	SW Test & Deployment	2011-11-07	2012-03-02
5	SUPPORT & MAINTENANCE (Waranty)	2012-03-05	2012-06-02

Tabela 5 – Planeamento de atividades do projeto *Green*

Como se pode constatar pela análise da Tabela 5, o desenvolvimento do projeto *Green* deveria ter concluído o desenvolvimento a 2 de março de 2012 e pronto para ser colocado em produção. A realidade foi bastante diferente do planeamento inicial. As atividades descritas de

análise e especificação, arquitetura e design, implementação e, testes e desenvolvimento foram realizadas em ciclo para cada *sprint* e todas sofreram um atraso significativo.

Devido ao atraso de desenvolvimento do projeto, existiram objetivos do estágio que não foram atingidos, uma vez que, comtemplavam tarefas realizadas no final do desenvolvimento do projeto, nomeadamente, a elaboração de documento de apoio para os utilizadores da aplicação.

Apesar das restrições de confidencialidade, será apresentada uma descrição e análise das tarefas e atividades realizadas para a implementação do projeto *Green*.

4.4. Operacionalização do plano de trabalho

Nesta secção sáo descritas e descriminadas diversas atividades e tarefas realizadas na operacionalização do plano de trabalho descrito no capítulo 3. Para uma melhor compreensão e estruturação, esta secção apresenta diversas subsecções:

- Definição de âmbito de trabalho;
- Gestão do projeto *Green;*
    - Metodologia de trabalho do projeto *Green;*
- Análise e testes;
- Avaliação de resultados e método de trabalho;
- Conclusões.

Na definição de âmbito de trabalho são mencionadas algumas alterações existentes a nível das equipas de trabalho e as consequências mais relevantes dessas alterações.

Na gestão de projetos descreve-se a forma como foram realizadas algumas fases, concepção e implementação, e a forma como foi estruturado o projeto. Ainda no âmbito da gestão de projeto é descrita a metodologia *scrum* aplicada ao projeto *Green*.

Posteriormente apresenta-se uma descrição sumaria da forma de realização da análise e de testes, seguindo-se uma avaliação dos resultados obtidos e dos métodos de trabalho aplicados.

4.4.1. Definição de âmbito de trabalho

O estágio curricular foi iniciado no fim de Setembro. Ao entrar para a empresa, a organização da empresa sofreu algumas alterações, nomeadamente a nível físico, uma vez que, existiu a adição de um lugar. Com o decorrer do tempo, e como a empresa se encontra em expansão, surgiram mais dois colaboradores que deram origem a mudanças estruturais na empresa. Inicialmente, a empresa estava estruturada por projeto numa única equipa. Existiam algumas funções realizadas somente por uma pessoa e outras que eram partilhadas por dois, ou mais elementos. Os projetos realizados na empresa tinham a participação dos diversos colaboradores da empresa.

Com a entrada de novos colaboradores, passaram a existir dois colaboradores por tipologia de competências, o que permitiu a criação de duas equipas de trabalho com competências redundantes.

Esta mudança estrutural provocou alterações positivas e negativas no desempenho de funções na empresa. Como mudança positiva encontra-se o facto de os projetos em curso passarem a estar divididos entre as duas equipas, permitindo uma melhor qualidade na concentração do trabalho individual de cada colaborador mas também como equipa. O facto de não terem de mudar o projeto em desenvolvimento com tanta frequência permite que sejam cometidos menos erros, devido aos colaboradores estarem mais focados no trabalho. Por outro lado, como o trabalho desenvolvido pela equipa de trabalho é programado com bastante detalhe, permite que os colaboradores estejam alinhados na realização de tarefas e permite a consulta e interação dos diversos elementos, sendo muitas vezes discutida a melhor estratégia a implementar para a realização de determinadas funções.

As alterações negativas, sentidas imediatamente após a constituição das equipas, consubstanciaram-se na perda de métodos de trabalhos existentes entres diversos colaboradores, que, devido ao facto de ficarem em equipas distintas ou ficarem a desempenhar funções diferentes das anteriores, não puderem manter a forma de trabalhar conquistada pela confiança e experiência de trabalho em grupo. Por outro lado, como umas das equipas ficou com dois elementos novos na empresa, teve uma dificuldade acrescida de integração e desenvolvimento de métodos de trabalho da equipa como um todo. Apesar dos prós e contras, verificou-se que as vantagens superam as desvantagens.

Após a reestruturação dos colaboradores em equipa, decorreu uma alteração na disposição física dos colaboradores, com o intuito de aproximar os elementos constituintes de cada equipa. Esta alteração proporcionou uma maior e melhor comunicação entre os elementos da equipa de trabalho, permitindo a alcance de melhores resultados na realização de projetos.

Cerca de cinco meses após o início do projeto, a equipa de desenvolvimento foi novamente reestruturada, perdendo um elemento, que passou a integrar uma nova equipa. Este facto provocou um atraso no desenvolvimento da aplicação sendo necessário definir que elementos iriam desempenhar as funções anteriormente atribuídas ao elemento que saiu e elaborar uma nova estratégia de desempenho de funções.

Relativamente ao projeto retratado neste relatório verificou-se que o gestor de projeto exerceu uma liderança elevada, sendo o elemento que manifestou mais poder e autoridade na elaboração do projeto. Verificou-se também que o gestor de projeto expressou uma grande dedicação do projeto e exerceu o controlo do orçamento do projeto. As caraterísticas deste projeto, tal como representado na Tabela 2 , adequam-se à organização por projeto, estrutura organizacional praticada pela *iUZ Technologies*. Tal como supramencionado no enquadramento teórico, este tipo de organização é adequada para a realização de projetos diferenciados e únicos como manifestado nos projetos desenvolvidos na empresa *iUZ Technologies*.

Em seguida, será abordada a gestão do projeto Green.

4.4.2. Gestão do projeto *Green*

A gestão de projeto de desenvolvimento de *software* supramencionada possui três fases principais: concepção, implementação e manutenção. Ao longo do estágio curricular no decurso do desenvolvimento do projeto *Green* acompanhamos duas dessas fases, a concepção e implementação. A fase de manutenção não será descrita pois, na data de término do estágio curricular, esta ainda não tinha ocorrido.

A fase de concepção, tal como mencionado no esquadramento teórico, deve identificar a informação que o projeto deve processar, as funcionalidades a implementar, as restrições existentes e os critérios que determinam o sucesso e a aceitação.

Antes da realização desta fase foram negociados com o cliente orçamentos, metas, prazos e objetivos que o projeto deveria cumprir. Após este trabalho, foi realizado um planeamento com as várias versões do projeto nas quais eram especificadas quais as funções que a aplicação deveria cobrir, em que *Work Package* estava incluída e em que data deveriam estar implementadas.

Na fase de concepção, foram realizadas no início do projeto diversas reuniões com o cliente e alguns elementos da equipa de desenvolvimento. Estas reuniões tiveram o intuito de enumerar, descrever e analisar os requisitos que se deviam implementar e funcionalidades que a aplicação deveria executar.

Após estas reuniões foi realizado um documento com todos os requisitos funcionais apurados. Neste documento foram descritos e especificados todos os atores intervenientes do projeto. Para facilitar o enquadramento de requisitos, estes foram agrupados em *Work Package* distintos, os quais identificavam as centrais funções dos principais atores.

Nos *Work Packages* destacavam-se os enumerados em seguida:

1- Serviços;
2- Administração;
3- Gestão de Oferta;
4- Gestão de Procura;
5- Meus serviços;
6- Canal de Distribuição *Green* .

Como alguns atores tinham funções de âmbito distinto, alguns Work Packages foram subdivididos em diversos *Work Packages*, como por exemplo, o WP4:

4.1 - Subdomínio
4.2 - *Widgets*
4.3 - API.

Para coadjuvar uma melhor compreensão geral do projeto, foram elaborados diagramas de atividade com a identificação das principais tarefas e funções a serem realizadas dentro de cada *work package*, identificando os respetivos intervenientes.

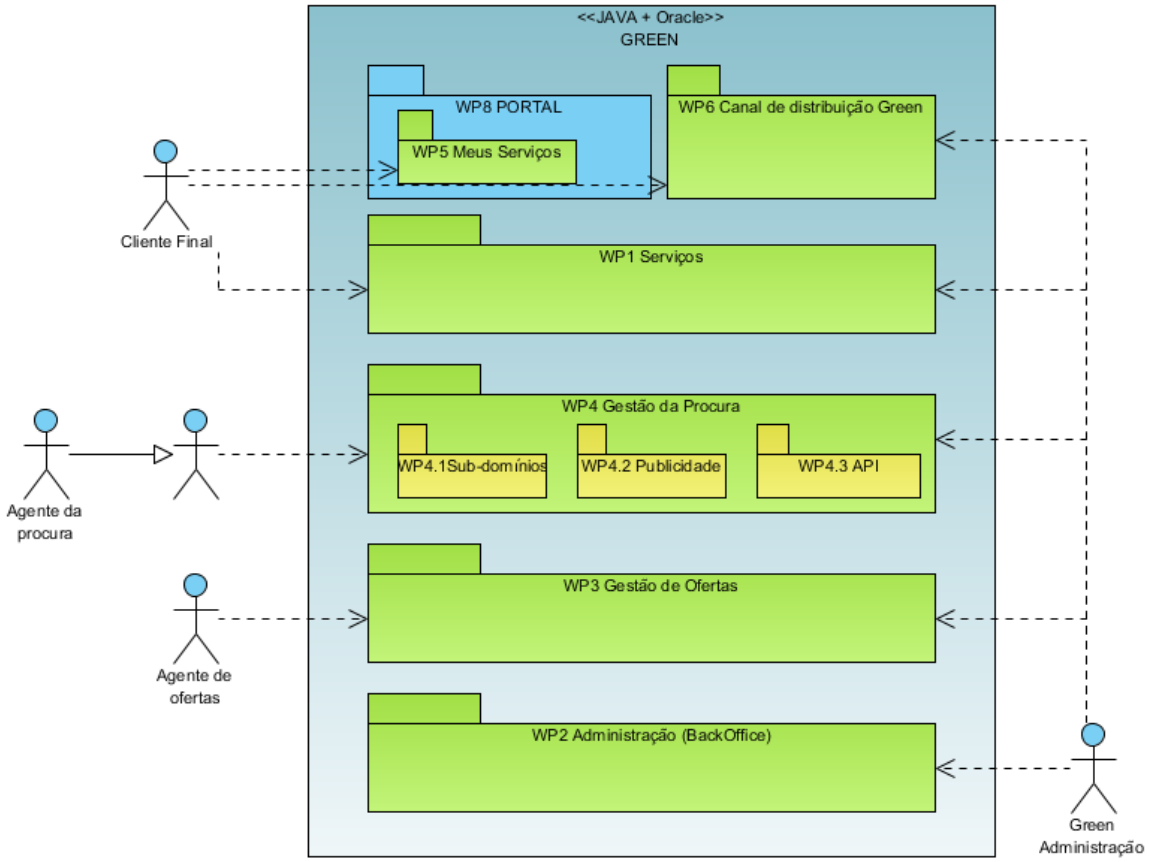


Figura 27 – Projeto *Green* com os atores e *Work Packages*

A fase de implementação da gestão de projeto tem o propósito de construir realmente o projeto, definindo e construindo as estruturas de dados, os programas, os módulos, as interfaces internas e externas e os testes a realizar

Para poder implementar a aplicação, os requisitos foram transformados em *user stories* autónomas, simples e funcionalmente válidas, através de um trabalho minucioso, com o intuito de poder realizar testes unitários na aplicação. Para facilitar o desenvolvimento e utilizando uma metodologia ágil, realizaram-se *sprints*, normalmente com 2 semanas de duração. A metodologia utilizada será apresentada em seguida.

4.4.2.1. Metodologia de trabalho do projeto *Green*

O projeto *Green,* descrito neste relatório, implementou uma metodologia de desenvolvimento de *software* ágil, denominada *Scrum.*

O projeto *Green* teve uma duração bastante elevada em relação ao previsto e diversas *sprints*. Na realização do projeto foram elaboradas diversas versões que foram entregues ao cliente regularmente para análise, correção e observação dos objetivos propostos. Habitualmente estas entregas eram realizadas ao final de uma *sprint* de desenvolvimento, na qual eram implementadas novas funcionalidades.

O desenvolvimento do projeto *Green* foi iniciado com o WP 1 – Serviços, por ser o ponto crucial do projeto. Este *work package* tinha como objetivo proporcionar ao cliente final, representante de todos os utilizadores com acesso à internet, a possibilidade de solicitar um serviço. Esta parte do projeto iria necessitar da configuração da oferta e da procura, ou seja, para funcionar corretamente iria ser necessário desenvolver todo o restante projeto. Para auxiliar o desenvolvimento, os serviços foram segmentados em tarefas que se encontravam estruturados em 5 páginas:

1. Pesquisa
2. Soluções
3. Serviços extras
4. Dados
5. Página de confirmação

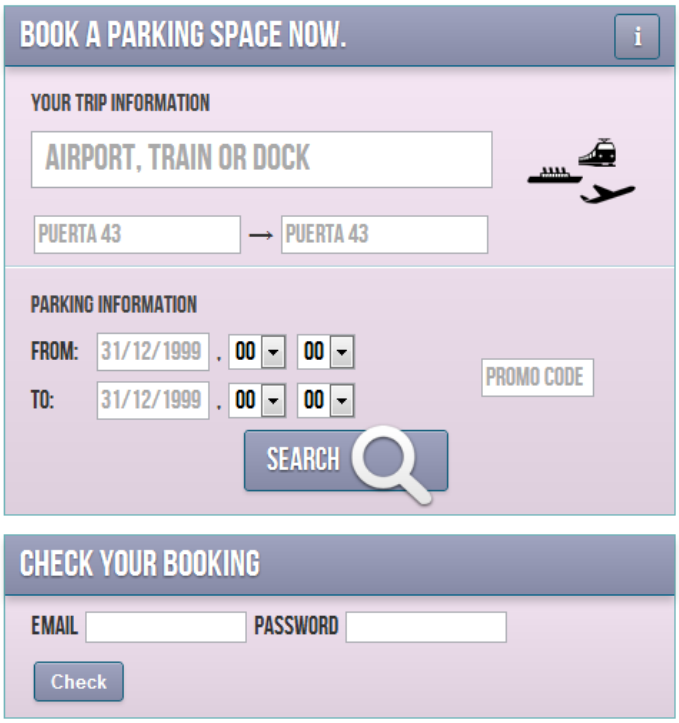


Figura 28 – Página 1: Pesquisa do Projeto Green

A página de pesquisa possibilitava o preenchimento a data, hora e local onde desejava que o serviço estivesse disponível (Figura 28) e posterior pesquisa das várias Soluções disponíveis.

Cada Solução continha uma descrição sucinta (Figura 29) e outra mais detalhada que poderia ser consultada, caso o cliente final desejasse no "+ info". Para além desta funcionalidade, o cliente poderia ordenar as soluções apresentadas, poderia filtrar os dados mediante a escolha de um tipo de serviço, mediante a cobertura e/ ou tendo em conta o tipo de caraterísticas que as diversas soluções ofereciam.

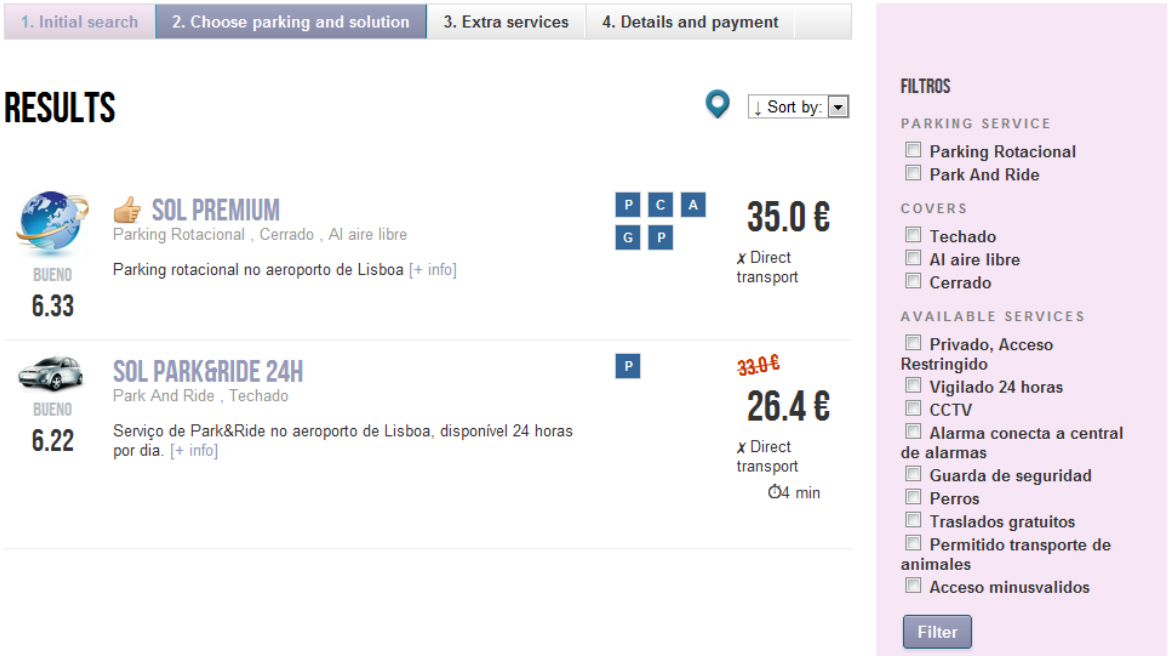


Figura 29 – Página 2: Soluções do Projeto *Green*

Em seguida eram apresentados vários serviços extras que poderiam ser adicionados à seleção inicial da solução. Depois era apresentado um formulário com vários dados que o cliente teria de preencher. Finalmente, chegaria a uma página de confirmação da reserva, que permitia a impressão do documento comprovativo e enviava um email para o cliente final com o comprovativo detalhado da reserva.

A primeira *sprint* não foi a mais complexa, mas devido à novidade do contexto do projeto e à introdução dos conceitos e requisitos à equipa de desenvolvimento, foi das mais demoradas. Um contexto novo aliado a uma equipa de trabalho diferente fez com que o desenvolvimento inicial fosse mais demorado que o restante desenvolvimento do projeto.

O procedimento de desenvolvimento de cada *sprint* esteve estruturado na elaboração de diversas tarefas e atividades que percorriam todo o ciclo de desenvolvimento da sprint. Estas tarefas e atividades foram bastantes semelhantes em todas a *sprints* realizadas e, por este motivo, bem como considerar informação redundante, iremos descrever em pormenor apenas uma *sprint*, neste caso, a *sprint* 8 dedicada às ofertas.

Esta *sprint* teve a duração de 16 de abril a 27 de Abril de 2012. Nesta *sprint*, o papel de *scrum master* esteve atribuído a mim.

Antes de iniciar a *sprint* foram realizadas algumas funções de elaboração e atualização das *user stories*, bem como, da atualização do protótipo. O protótipo foi enviado para o cliente para aprovação. O cliente aprovou-o e sugeriu algumas melhorias, no entanto, como estas sugestões foram enviadas no decorrer da *sprint*, tiveram de ser feitas algumas correções às funcionalidades já implementadas.

O primeiro dia da *sprint* foi dedicado ao planeamento. Esta reunião denominada *srum planning*, presidida pelo *product owner*, tinha como objetivo crucial planear a *sprint*, bem como, transmitir à equipa de desenvolvimento quais as funcionalidades a implementar, analisar o protótipo e definir qual a estratégia para desenvolver as funcionalidades. Nesta reunião foram analisadas, completadas e atualizadas as *user stories* que fazem parte desta *sprint*. Para cada *user story* foi feita uma estimativa do tempo que demoraria para implementar as funcionalidades para cada um dos elementos. Depois desta análise detalhada, por vezes, foram atualizados os prazos e/ou os objetivos a implementar no decorrer da *sprint*. Nesta *sprint* em particular foram alargados os prazos, uma vez que se entendeu que o período de uma semana era insuficiente para o cumprimento dos objetivos pretendidos. Após a realização do planeamento, a equipa de desenvolvimento iniciou o desenvolvimento do projeto pelas funções que haviam sido discutidas e planeadas na reunião.

Os dias desta *sprint* foram bastante rotineiros e representativos de grande parte dos dias de trabalho.

O dia iniciava com uma *daily scrum*, onde eram reunidos todos os elementos da equipa, incluindo o *scrum master* e o *product owner* quando este estava disponível. Nesta reunião era descrito o trabalho que tinha sido realizado no dia anterior por cada elemento da equipa, qual o trabalho que cada um planeava realizar nesse dia e era questionado se existia algum impedimento para o trabalho que iriam realizar nesse dia ou num futuro próximo. Quando existiam impedimentos, estes ficavam a cargo do *scrum master* e eram tomadas medidas para o resolver da forma mais rápida possível de forma a poder retirar o melhor rendimento de cada um dos colaboradores. Quando o *scrum master* não tinha possibilidades de resolver o impedimento, este era atribuído a um dos elementos da equipa, para que pudesse ser resolvido de forma breve e eficiente.

As reuniões da equipa de desenvolvimento do projeto *Green* eram realizadas próximo de um quadro que tinha as *user stories* que já haviam sido implementadas e que iriam ser desenvolvidas nessa *sprint*. O quadro tinha vários estados: *backlog*, onde constavam todas as *user stories* que iriam ser desenvolvidas nessa *sprint*; *in progress*, que continha todas as *user stories* que estavam em desenvolvimento; e *sprint done* que continha as *user stories* que já tinham sido desenvolvidas mas que ainda não tinham sido efetuados os testes de aceitação, como se pode observar na Figura 30. Após terem sido realizados os testes de aceitação, as *user stories*

transitavam para o estado *product done*. Durante a reunião, mediante a informação fornecida por cada elemento da equipa, as *user stories* eram manipuladas de um estado para outro e permitia ao *scrum master* e ao *product owner*, bem como, aos restantes elementos da equipa, ter uma noção bastante exata e pormenorizada da velocidade de desenvolvimento e percepcionar que funcionalidades faltavam implementar e quais já estavam desenvolvidas.

Figura 30 – Quadro do projeto *Green*

Fazendo uma comparação com as indicações apresentadas por Mike Cohn para a *daily scrum* (Mountain Goat Software, n. d.-a), na minha opinião, cumprimos as mesmas com elevada precisão.

No final da *sprint*, a *sprint review* e *sprint retrospective* foram suprimida, por falta de tempo para a sua realização e ausência do *product owner* na empresa na data prevista para a sua realização.

A *sprint review* e *sprint retrospective* foram realizadas em cerca de metade das *sprint*s. Normalmente fazia-se a *sprint review* seguida da *sprint retrospective*. Na primeira começávamos por ver as funcionalidades que tinham sido implementadas mas que ainda não tinham sido testadas. Após esta demonstração, realizávamos a *sprint retrospective* em que cada elemento do grupo de trabalho dava a sua opinião sobre a *sprint* que tinha decorrido e quais eram os motivos que consideravam responsáveis por não terem sido atingidos alguns objetivos das *sprint*s. Destas reuniões surgiram algumas medidas que foram implementadas de forma a melhorar o desempenho da equipa.

Uma das melhorias implementadas foi a elaboração de diagramas de estado e de sequência para as diversas funcionalidades, para que a equipa tivesse uma ideia clara, precisa e concreta dos diversos estados possíveis, do ciclo de evolução desses ciclos e mais tarde passaram a incluir também a informação sobre quais os atores que poderiam editar os dados e quais os destinatários que iriam receber emails de notificação de diversas ações. Encontram-se em anexo, 3 exemplos de diagramas de estado elaborados no âmbito do projeto *Green*, que mostra a versão

mais simplista em que apenas têm os estados e as ações que podem ser realizadas em cada estado até uma versão bastante complexa de diagrama em que constam todos os estados, as diversas ações, os emails que deverão ser enviados (Estado das ofertas). Estes exemplos podem ser consultados no Anexo 2.

Existiram outras melhorias como, por exemplo, incluir uma *user story* com a realização do protótipo das funcionalidades de *sprint* seguinte. Esta melhoria adveio do facto da equipa de trabalho considerar imprescindível ter o protótipo realizado e validado, de forma a serem realizadas o menor número de alterações e poderem consultar o mesmo para observarem as diversas ações que tinham de ser implementadas. Numa das *sprint*s, o protótipo ainda não estava desenvolvido e optou-se por iniciar o desenvolvimento das funcionalidades sem a elaboração do mesmo para poupar tempo. Esta ação não obteve o resultado esperado, uma vez que, devido a não terem o protótipo para consultar as funcionalidades, surgiram muitas dúvidas na implementação das funcionalidades. O facto de não terem uma ideia global de todas as tarefas a realizar, fez com que determinadas particularidades e restrições não estivessem contempladas, o que provocou a retificação de código e repetição do desenvolvimento de várias tarefas. Nesta *sprint* o desenvolvimento e implementação das funcionalidades teve uma velocidade bem mais reduzida que as demais, o que demonstrou que o protótipo era essencial para esta equipa de desenvolvimento. Este facto era corroborado por, na altura do desenvolvimento desta *sprint*, muitos dos requisitos não estarem registados num documento. Este foi, na minha opinião, um dos grandes problemas do desenvolvimento do projeto.

4.4.3. Análise e testes

Na fase de análise, foi elaborado um documento com os requisitos funcionais que o projeto *Green* deveria realizar, como supramencionado. Para o desenvolvimento do projeto, os requisitos foram transformados em *user stories*. No início de cada *sprint*, as *user stories* eram colocadas no quadro ilustrado na Figura 30 e transitavam de estado à medida que o desenvolvimento decorria.

As *user stories*, exemplificadas na Figura 31 foram eleitas como o meio de transmitir os requisitos à equipa de trabalho que iria realizar o desenvolvimento do projeto *Green*. Para além de permitir a manipulação de *user stories* de modo a proporcionar uma visão global do desenvolvimento do projeto aos vários elementos da equipa de desenvolvimento e ao *product owner*, tinham pequena dimensão, eram claros e precisos, enfatizam a comunicação verbal permitindo estimativas de dificuldade e tempo que demoravam a desenvolver, eram suficientemente pequenas pelo que permitiam ser desenvolvidas várias na mesma sprint, permitem flexibilidade na ordem do seu desenvolvimento e acima de tudo, são testáveis.

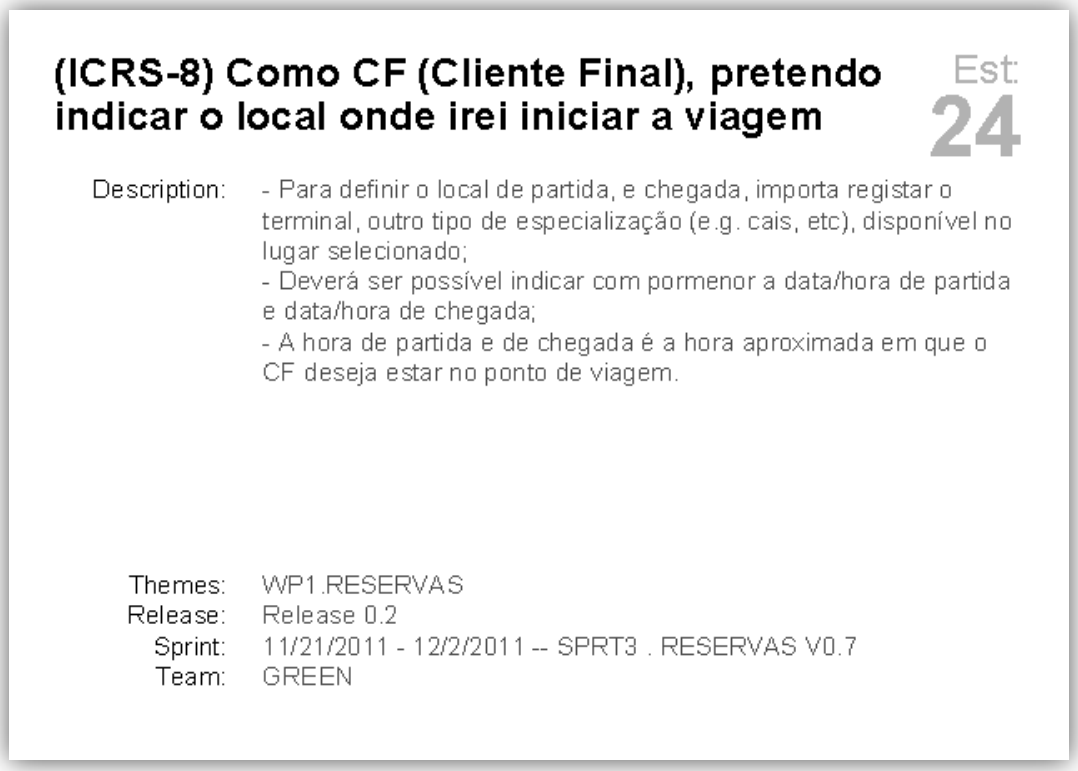


Figura 31 – Exemplo de uma *user story* aplicada no projeto *Green*

Para testar cada *user story*, esta era copiada do programa onde era elaborada e armazenada, *Scrum Works*, para um programa denominado *Eventum*. Neste programa, era descrita a user storie e descritos alguns testes que tinham de ser realizados. Muitas das *user stories* estavam interligadas e por este motivo, ao realizar os testes, o objetivo era realizar os testes por *epic*, ou seja, o conjunto de *user stories* que completavam uma funcionalidade. Esta forma de realizar os testes permitia que se poupasse bastante tempo na elaboração de pre-requisitos e caraterísticas necessárias para a realização de testes. Quando existia algum erro ou melhoria a implementar, este era descrito com muito pormenor e endereçado a um dos elementos da equipa de trabalho que, efetuavam a sua correção. Cada *issue* encontrado, para além de estar incluído habitualmente numa categoria de erro ou melhoria e de ser endereçado a um ou vários elementos da equipa de trabalho, possuía a classificação por prioridade: crítica, alta, média, baixa e agradável possuir. Esta última categoria, agradável possuir era destinada a registar as melhorias que poderiam ser implementadas para contribuir para a excelência do projeto mas que, no entanto, não pertenciam aos requisitos discriminados pelo cliente.

Após as correções dos erros, era novamente verificado se o problema detetado já estava corrigido e, em caso afirmativo, era encerrado o *issue* e colocada a *user story* como realizada. Neste momento, a *user story* alcançava o estado de *done product* no quadro do projeto *Green*, objetivo pretendido para todas as *user stories*.

4.4.4. Avaliação de resultados e método de trabalho

Como referido anteriormente, uma dos maiores problemas do desenvolvimento do projeto *Green*, foi a inexistência de um documento que contemplasse o registo de todos os requisitos que o projeto deveria cumprir.

Quando o projeto iniciou, foi realizado um documento com as especificações dos requisitos. Nesse documento constavam cerca de 200 requisitos. Esse documento que inicialmente parecia bastante exaustivo demostrou estar muito incompleto em diversas áreas e funcionalidades. Por decisão do *product owner* e de acordo com a filosofia *scrum* de que o planeamento é iterativo (Dybå & Dingsøyr, 2008), iniciou-se o desenvolvimento do projeto sem o registo de todos os requisitos do projeto. Esta decisão, como todas as demais, teve vantagens e desvantagens. Uma das grandes vantagens foi a poupança de tempo e trabalho na recolha pormenorizada de todos os requisitos. Este trabalho era muito moroso e algo enfadonho para o cliente. A metodologia aplicada consistia em começar a desenvolver o projeto, e à medida que este ia sendo desenvolvido, ia-se falando com o cliente das próximas funcionalidades a desenvolver. Como a metodologia implementada (*scrum*) era ágil, permitia que a equipa de desenvolvimento se adaptasse aos novos requisitos. As grandes desvantagens desta decisão foram a alteração de requisitos fundamentais que originaram muitas mudanças no código, os programadores não conseguirem prever várias situações por não terem uma visão global de todas as funcionalidades do projeto e o desconhecimento e alteração de requisitos durante ou após a implementação de funcionalidades. Estes motivos, na minha opinião, provocaram um decréscimo na velocidade de desenvolvimento do projeto.

O desenvolvimento do projeto foi também afetado por outros fatores, uns dependentes da decisão da equipa de planeamento e desenvolvimento e outros alheios à mesma. Existiram *sprints* em que, devido à falta de tempo para realizar as tarefas, optou-se pelo desenvolvimento da aplicação sem a realização do protótipo. Apesar de inicialmente parecer que se poupou tempo, na realidade verificou-se que isto não sucedeu. Surgiu um número muito mais elevado de questões e parte do trabalho realizado teve de ser refeito devido a más interpretações dos objetivos requeridos. A equipa de desenvolvimento teve dificuldades acrescidas por não terem uma ideia clara e precisa dos objetivos e das funcionalidades que deveriam implementar.

Por vezes os elementos das equipas de desenvolvimento eram solicitados para a participação em outros projetos da empresa, o que também provocou alguma oscilação nas metas e prazos das *sprints* que estavam a ser realizadas.

No decorrer da implementação do projeto, realizaram-se diversas alterações de requisitos. Observaram-se várias reações como o aumento da duração do projeto e dos custos inerentes ao mesmo, aumento de erros que não tinham sido detetados e erros em requisitos pois as alterações introduzidas entravam em conflito com alguns que já estavam implementados. Para além destas reações, na equipa de trabalho observou-se uma diminuição da motivação dos trabalhadores e uma excessiva pressão para cumprir os objetivos estabelecidos. Na minha

opinião, não ocorreu uma diminuição de produtividade mas os demais efeitos provenientes da alteração de requisitos, identificados por Ferreira et al. (2009) ocorreram todos. Podemos concluir que a volatilidade dos requisitos tem vários efeitos indesejados para a implementação do projeto.

Outro problema advindo da alteração de requisitos e também da correção de erros, foi o surgimento de erros e falhas em funcionalidades e *user stories* que já estavam testadas. Diversos erros foram detetados provenientes dessas ações mas, para que a *iUZ Technologies* seja reconhecida pela qualidade e excelência recomenda-se o teste de todas as user stories implementadas antes de este ir para produção.

Como referido no enquadramento teórico, para sobreviver, prosperar, e crescer, uma metodologia deve ser capaz de aprender com outras metodologias e adaptar com sucesso a novas exigências do mercado (Riehle, 2001). Na minha opinião, seria interessante e produtivo implementar algo como o referido na metodologia Método Dynamic Systems Development Methodology (DSDM) (Voigt, 2004) em que constituíam um conjunto de requisitos imutáveis, que permaneciam ao longo de todo o desenvolvimento sem alterações. Desta forma, era permitida a alteração de requisitos, mas não na sua totalidade, trazendo um pouco de estabilidade ao desenvolvimento do projeto e permitindo ter uma metodologia ágil.

Na minha opinião, uma *sprint* de desenvolvimento "ideal" para este projeto, deveria ter duração de 2 semanas. Iniciaria com uma *sprint review* em que seriam esclarecidos, detalhados e planificados os objetivos dessa *sprint*. Durante a *sprint* deveria ser realizado o desenvolvimento das tarefas e implementadas as funções determinadas como objetivos. Todos os dias de trabalho posteriores ao primeiro, deveriam ser iniciados com uma daily scrum, onde seria realizado um ponto de situação com cada um dos trabalhadores e verificado a existência de impedimentos para a realização de trabalho.

Além destas atividades, deveria ser reunida a informação relativa às próximas funcionalidades a implementar para se poder realizar o protótipo das mesmas. Este protótipo deveria ser validado pelo *product owner* e pelo cliente final, se fosse necessário. Desta forma quando se iniciasse a *sprint* seguinte iria ter-se um nível de segurança satisfatório das tarefas a implementar e quais as restrições e requisitos dos mesmos. Depois desta validações deveriam ser realizadas e atualizadas as *user stories* que iriam ser desenvolvidas na *sprint* seguinte.

Outra tarefa que deveria ser iniciada no princípio de cada *sprint*, no meu ponto de vista, é a realização de testes de aceitação e de qualidade sobre as tarefas implementadas na *sprint* anterior. Desta forma, os programadores têm presente as caraterísticas das funcionalidades e a forma como estas foram implementadas e permite que a correção de erros seja realizada mais rapidamente e com melhores resultados.

No final da *sprint* deveria ser realizada uma *sprint review* e, se possível, uma *sprint retrospective*, segundo as indicações presentes no enquadramento teórico (Mountain Goat Software, n. d.-c, n. d.-d). Deve-se ter em conta que o desenvolvimento de funcionalidades só

esta concluído após a realização de testes de aceitação e da correção de erros que surjam na realização desses testes.

Sprint	
1ª Semana	**2ª Semana**
Sprint Planning	
Desenvolvimento de funcionalidades	
	Sprint review *Sprint retrospective*
Realização testes de aceitação da *sprint* anterior	
Realização do protótipo da *sprint* seguinte	Validação do protótipo da *sprint* seguinte
	Preparação e retificação das *user stories* da *sprint* seguinte

Figura 32 – Atividades de uma *sprint* "ideal"

4.5. Conclusões

Em suma, o trabalho realizado na *iUZ Technologies* foi muito gratificante para mim, aprendi imensos conceitos novos e fui introduzida numa área que me era praticamente desconhecida. A participação no projeto *Green* desde o seu início permitiu-me perceber a metodologia utilizada na empresa, quais os seus pontos fortes e aspetos que poderiam ser melhorados.

Ao longo de todo o projeto verificou-se a utilização da metodologia ágil *scrum* que seguia grande parte das indicações do enquadramento teórico. Iniciava-se a *sprint* com uma *sprint planning* em que eram especificadas as funcionalidades a implementar, esclarecidas dúvidas e feito o planeamento detalhado das tarefas. Os restantes dias da *sprint* eram principiados com uma *daily scrum* em que se sumariava o trabalho realizado por cada elemento de trabalho no dia anterior e qual o trabalho que iria ser realizado no dia seguinte. Quando surgiam impedimentos estes eram descritos para que o *scrum master* pudesse tomar providencias para os eliminar ou delegar essa responsabilidade noutro elemento.

A *sprint review* e *retrospective* foram realizadas apenas algumas vezes e pelo menos a *sprint review* tem, na minha opinião, uma importância elevada para o bom funcionamento da equipa de trabalho e não deveria ser descurada. Esta reunião permite que toda a equipa esteja sempre atualizada nas funcionalidades implementadas no desenvolvimento do projeto.

Para além destas indicações é proposta a elaboração de um conjunto de requisitos imutáveis, permitindo que estes servissem de base para a realização do projeto. Este conjunto

constituiria apenas uma parte dos requisitos melhorando a projeção da arquitetura geral do projeto, permitindo a alteração dos restantes requisitos. No entanto, esta alteração deverá ser ponderada, porque a alteração ocorrida nos requisitos neste projetos provocou vários efeitos secundários não desejados. Para além da satisfação imediata do cliente com a alteração de requisitos, o aumento do tempo na realização das tarefas e o custo advindos da alteração de requisitos, por exemplo, provocam uma diminuição da sua satisfação.

Recomenda-se, no final da fase de testes, a realização de testes de aceitação em todas as *user stories* com a finalidade de corrigir e colmatar o máximo de falhas, umas ainda não detetadas e outras provenientes de outras alterações no sistema, para que o projeto aufira da melhor qualidade e excelência.

Para finalizar, são apresentadas as atividades "chave" consideradas para a realização da *sprint* "ideal" para este projeto. Nestas atividades encontram-se as tarefas da *sprint* atual: *sprint planning*, *daily sprint*, desenvolvimento das funcionalidades, *sprint review* e *sprint retrospective*. Para além destas atividades, devem ser realizadas ações para um planeamento atempado da *sprint* seguinte: construção e validação do protótipo. Devem, também, ser realizados os testes logo após a implementação das funcionalidades pois permitem uma correção melhor e mais rápida dos erros detetados, uma vez que, os programadores e designers têm a informação e particularidades destas funções muito presentes.

No capítulo seguinte serão apresentadas as conclusões gerais de todo o relatório.

5. Conclusão

O estágio curricular realizado na *iUZ Technologies* foi muito gratificante para mim, aprendi imensos conceitos novos e fui introduzida numa área que me era praticamente desconhecida. A participação no projeto *Green* desde o seu início permitiu-me perceber a metodologia utilizada na empresa, quais os seus pontos fortes e aspetos que poderiam ser melhorados.

Neste relatório, o enquadramento teórico apresentado centra-se em 3 grandes temáticas: organização em equipas e em projeto, modelos de desenvolvimento de *software* e, análise e testes, temáticas que são analisadas no desenvolvimento do projeto *Green*.

Relativamente à organização da empresa concluímos que esta se encontra organizada por projeto, o que é adequado, uma vez que, na empresa são desenvolvidos projetos únicos e diferenciados. O ciclo de vida de um projeto pode ser bastante diferente e deve adequar-se às caraterísticas e diversidade do projeto. Consideramos que o ciclo de vida do projeto *Green* apresenta 7 fases distintas: iniciação, planeamento, desenvolvimento, testes, correção de bugs e verificação, exploração e finalmente, o encerramento.

O modelo de desenvolvimento de *software* do projeto *Green* utilizou uma metodologia ágil, mais especificamente a *scrum*. Este método é organizado em *sprints* e prevê entregas faseadas do projeto. Neste método seguiam-se grande parte das indicações presentes no enquadramento teórico. Recomenda-se para uma melhoria de resultados a realização de *sprints review* e *retrospective,* principalmente da *sprint review* no final da *sprint* com o intuito de manter toda a equipa atualizada nas funcionalidades implementadas no desenvolvimento do projeto. No desenvolvimento da *sprint* aconselha-se o desenvolvimento e validação do protótipo com as funcionalidades a implementar na sprint seguinte, bem como, a realização de testes de aceitação logo após a implementação das funcionalidades.

Relativamente à analise e testes, os requisitos foram transformados em *user stories* e toda a implementação das funcionalidades e desenvolvimento de tarefas foi baseada nas *user stories*. Relativamente aos testes, cada *user story* era testada independentemente de forma a garantir que todas as funcionalidades pretendidas estavam corretamente implementadas. Este método de realizar os testes é considerado por vários autores o que obtém melhores resultados. Além destas caraterísticas enumeradas é recomendada a repetição da realização de testes de aceitação em todas as *user stories* implementadas antes do projeto ser colocado em produção para alcançar uma maior qualidade e excelência do produto.

5.1. Limitações e propostas para trabalhos futuros

Nas limitações encontradas neste projeto destaca-se o facto de o projeto *Green* não ter sido concluído por motivos que são alheios à autora e desta forma não ter sido possível atingir todos os objetivos que tinham sido propostos inicialmente.

Outra limitação, que teve bastante impacto neste relatório, foi a privacidade e normas de segurança da empresa que não permitiram uma descrição clara e pormenorizada das diversas tarefas e atividades realizadas no âmbito do projeto *Green*, nem uma descrição pormenorizada do projeto, tendo sido efetuada uma descrição de forma um tanto ou quanto anonimizada neste relatório, sem que isso prejudicasse quer a empresa de acolhimento, quer os objetivos deste trabalho.

Como trabalho futuro, penso que seria interessante desenvolver um novo projeto tendo em conta as melhorias sugeridas no capítulo anterior, terminar o projeto *Green* e realizar uma comparação entre as duas metodologias aplicadas, para verificar se as melhorias sugeridas iriam surtir o efeito desejado e que novas medidas deveriam ser implementadas no desenvolvimento de novos projetos na *iUZ Technologies*.

5.2. Balanço final do estágio curricular

Fazendo um balanço geral do estágio curricular, tendo em conta os objetivos iniciais propostos, todos foram atingidos com a exceção da elaboração da documentação de apoio aos utilizadores. Este objetivo não foi cumprido porque a realização do projeto *Green* teve atrasos consideráveis no seu desenvolvimento, face ao inicialmente estimado e, na altura do término de estágio, ainda não se tinha alcançado o momento de realizar esta tarefa. Todos os restantes objetivos foram alcançados com sucesso.

Na minha opinião, o estágio curricular realizado na *iUZ Technologies* foi vantajoso para a empresa, pois permitiu usufruírem dos meus serviços e para mim porque possibilitou um enriquecimento a nível pessoal e profissional bem como uma vasta e rica aprendizagem.

Ao longo do decurso do estágio curricular, com a noção que o protótipo era essencial para a equipa de desenvolvimento, passámos a realizar e analisar o protótipo com muito mais ênfase e atenção que anteriormente. Habitualmente condensava todas as funcionalidades e tarefas que o protótipo deveria realizar e, à medida que este ia sendo desenvolvido, realizava uma análise para verificar se todas as funcionalidades estavam contempladas e sugerir algumas melhorias. Este trabalho permitia colmatar várias falhas que poderiam ter impacto no devenvolvimento do projeto.

Ao longo do estágio curricular adquiri diversos conceitos, metodologias, práticas e experiência. A grande mais-valia de um estágio relativamente ao estudo teórico é a experiência e possibilidades de implementar realmente os procedimentos adquiridos. Esta experiência foi ainda mais rica que o normal porque acompanhei o desenvolvimento do projeto *Green* desde o início, percorrendo diversas fases. A equipa de desenvolvimento deste projeto, bem como o *product owner*, eram experientes e sempre me auxiliaram e realizaram sugestões de melhoria do meu trabalho para que pudesse integrar-me ainda melhor nas minhas tarefas e atividades.

Na minha opinião, este estágio e todos os conhecimentos adiquiridos são muito importantes para o meu futuro e possuem uma grande possibilidade de expandir os meus horizontes ao nível do mercado de trabalho.

Fazendo uma avaliação do estágio, considero que este superou as minhas expectativas, pois não tinha considerado adiquirir tantos conhecimentos, experiência e trabalhar com os fantásticos colegas da *iUZ Technologies*. Outras caraterísticas que me surpreenderam foi a responsabilidade e poder adquirida no desenvolvimento do projeto, uma vez que, grande parte do desenvolvimento do projeto foi realizada baseada no trabalho elaborado por mim, ainda que este trabalho fosse verificado pelo *product owner*. Muitas vezes recorriam a mim para o esclarecimento de dúvidas relativas aos requisitos descritos ou às funcionalidades a implementar, sendo minha a responsabilidade de procurar o esclarecimento no *product owner* ou reunir a informação para solicitar o esclarecimento ao cliente. Como era o elemento do projeto que tinha mais conhecimento dos requisitos e havia participado na maioria das reuniões com o cliente era considerada o "cérebro" do projeto, pelo que alcancei importância no desenvolvimento do projeto.

6. Referências bibliográficas

Agile Alliance. (2001). Twelve Principles of Agile Software Retrieved 18/03/2012, from http://agilemanifesto.org/

Beck, K., & Fowler, M. (2000). *Planning Extreme Programming* (1st edition ed.). Boston: Addison Weley.

Boehm, B., & Basili, V. R. (2001). Software Defect Reduction Top 10 List. *Software Management*.

Bouabana-Tebibel, T., & Belmesk, M. (2007). An object-oriented approach to formally analyze the UML 2.0 activity partitions. *Information and Software Technology, 49*(9-10), 999-1016. doi: 10.1016/j.infsof.2006.10.007

Churchill, N., & Lewis, V. (1983). The Five Stages Of Small Business Growth. *Harvard Business Review*.

Cockburn, A. (2001). *Agile Software Development*: The Agile Software Development Series.

Coelho, A., Lisboa, J., Coelho, F., & Almeida, F. (2004). *Introdução à gestão das organizações*. Barcelos: Grupo Editorial Vida Económica.

Cohn, M. (2004). Advantages of User Stories for Requirements. *Informit Network*.

Costa, S. (2010). Especificação de requisitos e testes de usabilidade no Sistema de Informação para a Saúde Oral - SISO. *Universidade de Aveiro*.

Duncan, W. (2008). *A Guide to the Project Management Body of Knowledge* (4nd ed.): Project Management Institute

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9-10), 833-859. doi: 10.1016/j.infsof.2008.01.006

Ferreira, S., Collofello, J., Shunk, D., & Mackulak, G. (2009). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. *The Journal of Systems and Software, 82*, 1568-1577.

Gallegos, F., Senft, S., Manson, D. P., & Gonzales, C. (2004). *Information Technology Control and Audit* (Second Edition ed.): Auerbach Publications.

Hanssen, G. K., & Fægri, T. E. (2008). Process fusion: An industrial case study on agile software product line engineering. *Journal of Systems and Software, 81*(6), 843-854. doi: 10.1016/j.jss.2007.10.025

Haughey, D. (n. d.). The Project Management Body of Knowledge (PMBOK). *Project Smart*.

Highsmith, J., & Consortium, C. (2002). What Is Agile Software Development? *Agile Software Development*.

Hitt, M., Ireland, R. D., & Hoskisson, R. (2008). *Administração Estratégica* (E. G. Kanner, M., Trans. 2ª Edição ed.). São Paulo: Thomson Learning.

Hofmeister, C., Kruchten, P., Nord, R. L., Obbink, H., Ran, A., & America, P. (2007). A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software, 80*(1), 106-126. doi: 10.1016/j.jss.2006.05.024

Idani, A., & Ledru, Y. (2006). Dynamic graphical UML views from formal B specifications. *Information and Software Technology, 48*(3), 154-169. doi: 10.1016/j.infsof.2005.03.008

IIBA. (n. d.). The Guide to the Business Analysis Body of Knowledge™. *International Institute of Business Analysis*.

iUZ Technologies. (2011). iUZ Technologies Retrieved 10/10/2011, 2011, from www.iuz.pt

iUZ Technologies. (n. d.-a). Bike Tour Online Experience Retrieved 19/10/2011, from http://www.worldbiketour.net/

iUZ Technologies. (n. d.-b). J. F. Luso Retrieved 19/10/2011, from http://www.jfluso.pt/

iUZ Technologies. (n. d. -a). AGAP - Associação de Empresas de Ginásios e Academias de Portugal Retrieved 27/12/2011, from http://www.agap.pt

iUZ Technologies. (n. d. -b). Clube Atlântico da Madalena Retrieved 27/12/2011, from http://www.atlanticodamadalena.pt/

iUZ Technologies. (n. d. -c). Cymbolex - Isolamentos e Aquecimentos Unipessoal, Lda Retrieved 27/12/2011, from http://www.cymbolex.pt/

Karimi, A., & Noori, A. (2011). Software Engeneering and Enterprise Architecture - A comparison study. *International Journal of Academic Research, 3*.

Laleau, R., & Polack, F. (2008). Using formal metamodels to check consistency of functional views in information systems specification. *Information and Software Technology, 50*(7-8), 797-814. doi: 10.1016/j.infsof.2007.10.007

Lucas, F. J., Molina, F., & Toval, A. (2009). A systematic review of UML model consistency management. *Information and Software Technology, 51*(12), 1631-1645. doi: 10.1016/j.infsof.2009.04.009

Mariani, A. (n. d.). Conceito Básicos da Teoria de Grafos Retrieved 16/01/2012, from http://www.inf.ufsc.br/grafos/definicoes/definicao.html

Meredith, J., & Mantel, S. (2010). *Project Management: a Managerial Approach* (Seventh ed.): Wiley.

Microsoft. (2008). The IT Service Lifecycle Retrieved 12/03/2012, from http://technet.microsoft.com/en-us/library/cc543217.aspx

Miguel, A. (2003). *Gestão de Projetos de Software*. Lisboa: FCA - Editora de Informática.

Milunsky, J. (2009). Top 10 Activities of the Product Owner, from http://agilesoftwaredevelopment.com/blog/jackmilunsky/top-10-activities-product-owner

Miqdadi, A. (n. d.). *The 9 knowledge Areas and the 42 ProcessesBased on the PMBoK® 4th*.

Miranda, E., & Bourque, P. (2010). Agile monitoring using the line of balance. *Journal of Systems and Software, 83*(7), 1205-1215. doi: 10.1016/j.jss.2010.01.043

Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software, 82*(11), 1869-1890. doi: 10.1016/j.jss.2009.05.052

Mountain Goat Software. (n. d.-a). The Daily Scrum Meeting, from http://www.mountaingoatsoftware.com/scrum/daily-scrum

Mountain Goat Software. (n. d.-b). Learning Scrum - Free to Use Figures and Wallpapers from http://www.mountaingoatsoftware.com/scrum/figures

Mountain Goat Software. (n. d.-c). Sprint Retrospective, from http://www.mountaingoatsoftware.com/scrum/sprint-retrospective

Mountain Goat Software. (n. d.-d). Sprint Review Meeting. *Mountain Goat Software*, from http://www.mountaingoatsoftware.com/scrum/sprint-review-meeting

Nambisan, S. (2002). Software firm evolution and innovation–orientation. *Engineering and Technology Management, 19*, 141–165.

Ovaska, E., Evesti, A., Henttonen, K., Palviainen, M., & Aho, P. (2010). Knowledge based quality-driven architecture design and evaluation. *Information and Software Technology, 52*(6), 577-601. doi: 10.1016/j.infsof.2009.11.008

Pino, F. J., Pedreira, O., García, F., Luaces, M. R., & Piattini, M. (2010). Using Scrum to guide the execution of software process improvement in small organizations. *Journal of Systems and Software, 83*(10), 1662-1677. doi: 10.1016/j.jss.2010.03.077

PMBOK Guide 4th Edition Process Groups Knowledge Areas Mapping.

Pressman, R. (2000). Software Engineering: A Practitioner's Approach: McGraw-Hill.

Priberam. (2010). Dicionário Priberam da Língua Portuguesa Retrieved 2012-06-04, from http://www.priberam.pt/DLPO/default.aspx?pal=software

Qumer, A., & Henderson-Sellers, B. (2008a). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology, 50*(4), 280-295. doi: 10.1016/j.infsof.2007.02.002

Qumer, A., & Henderson-Sellers, B. (2008b). A framework to support the evaluation, adoption and improvement of agile methods in practice. *Journal of Systems and Software, 81*(11), 1899-1919. doi: 10.1016/j.jss.2007.12.806

Qureshi, M., & Hussain, S. A. (2008). An adaptive software development process model. *Advances in Engineering Software, 39*.

Riehle, D. (2001). A Comparison of the Value Systems of Adaptive Software Development and Extreme Programming: How Methodologies May Learn from Each Other. *eXtreme Programming Examined*.

Royce, W. (1970). Managing the development of large software systems. *The Institute of Electrical and Electronics Engineers*.

Silva, A., & Videira, C. (2001). *UML, Metodologias e Ferramentas CASE* (1 ed.): Centro atlântico.

Stair, R., & Reynolds, G. (2005). *Princípios de sistemas de informação: uma abordagem gerencial* (6ª ed.). Brasil: Thomson.

Suscheck, C. (2012). Defining Requirement Types: Traditional vs. Use Cases vs. User Stories. *Agile Journal*.

UATEC. (2011). iUZ Technologies Retrieved 10/10/2011, from http://www.ua.pt/uatec/PageText.aspx?id=7285

Voigt, B. (2004). Dynamic System Development Method. *Department of Information Technology, University of Zurich*.

Weiglhofer, M., Fraser, G., & Wotawa, F. (2009). Using coverage to automate and improve test purpose based testing. *Information and Software Technology, 51*(11), 1601-1617. doi: 10.1016/j.infsof.2009.06.008

Wiegers, K. E. (2006). *More about software requirements: Thorny Issues and Practical Advice* Microsoft Press.

ANEXO 1

Ao longo do estágio fui realizando diversas atividades e tarefas na empresa. Segue-se uma síntese das tarefas mais importantes realizadas, ao longo do tempo.

Semana	Intervalo de dias	Descrição das tarefas
1	26-09-2011 a 02-10-2011	Nesta primeira semana, foi-me apresentada a empresa. Realizei a leitura de vários documentos relativos ao modo de funcionamento da empresa para me introduzir no modo de trabalho e funcionamento da empresa. Iniciei, também, a leitura do livro de UML – metodologias e ferramentas case (Silva & Videira, 2001) . Este livro foi-me recomendado porque possuia informação detalhada sobre uma série de conceitos que eu iria necessitar e permitia-me relembrar uns e tomar conhecimento de outros. O livro contém um capítulo com enquadramento e conceitos gerais que abrange os sistemas de informação, a arquitetura de sistemas de informação, o planeamento estratégico de sistemas de informação e engenharia de *software*, entre outros. O livro explica o processo de desenvolvimento de *software* que possui a descrição das várias fases de desenvolvimento de *software*, o que foi muito importante para me situar no processo e perceber qual a minha função e que impacto teria ao longo do projeto. Silva e Videira (2001) possuem uma parte do livro dedicada à linguagem de modelação UML, a qual é a utilizada pela empresa iUZ, a qual foi muito importante para a perceção de muitos conceitos e para entender a especificação, construção, visualização e documentação de artefactos de um sistema de informação.
2	03-10-2011 a 09-10-2011	Nesta semana realizei alguma pesquisa sobre a *iUZ Technologies*, nomeadamente, através da análise do *site* da empresa (iUZ Technologies, 2011). No entanto, a tarefa que realizei a grande parte do tempo foi relativa ao projeto de Auditoria. Realizei um estudo sobre este projeto para percepcionar o âmbito do projeto, as suas especificações e requisitos. Este projeto encontra-se me fase de manutenção e iniciei a realização de testes de desenvolvimento.

Semana	Intervalo de dias	Descrição das tarefas
3	10-10-2011 a 16-10-2011	Nesta semana foi realizada a continuação de testes de desenvolvimento do projeto de Auditoria. Nesta semana foi, também, iniciado o projeto *Green*, no qual é focado este documento e descrito todo o processo. O primeiro passo realizado por mim foi o levantamento de requisitos funcionais, o que devido à dimensão do projeto, foi apenas o início da realização do mesmo.
4	17-10-2011 a 23-10-2011	No âmbito do projeto *Green*, foram realizadas diversas reuniões com o cliente e com a equipa de trabalho com o objetivo de fazer o levantamento das necessidades do cliente, a sua alteração e atualização – Requisitos. Os requisitos foram agrupados em Work Packages para, entre outros objetivos, estruturar os dados e facilitar a sua consulta.
5	24-10-2011 a 30-10-2011	Nesta semana continuaram as reuniões com o cliente do projeto *Green* e a posterior alteração e correção de requisitos. Para além do levantamento de requisitos, colaborei na definição de *uses cases*, apesar de ainda ser uma versão inicial que não abrangia o projeto na sua totalidade. Nos últimos dias participei no início da elaboração do documento que contém as *user stories*.
6	31-10-2011 a 06-11-2011	Neste período, continuei com a alteração e atualização de requisitos, do projeto *Green*. Realizei também a atualização de *uses cases* e de *user stories*, de acordo com as alterações realizadas nos requisitos. Nesta semana foi realizada uma mudança estrutural no projeto *Green*, que implicará a retificação e reestruturação de grande parte do projeto.
7	07-11-2011 a 13-11-2011	Continuei com a realização e correção de *uses cases* e de *user stories*. Realizei uma pequena pesquisa para perceber como realizar o plano de testes baseados nas *user stories* já realizadas, no entanto, esta não foi suficiente para esclarecer as minhas dúvidas e perceber qual a forma mais ágil e competitiva de realizar os mesmos. Iniciei a leitura do livro *PMBOK - A Guide to the Project Management Body of Knowledge 4th* (Duncan, 2008), que explica como gerir um projeto e, nomeadamente, definem projeto, gestão de projeto e descrevem o ciclo de vida do projeto e a sua organização.

Semana	Intervalo de dias	Descrição das tarefas
8	14-11-2011 a 20-11-2011	As *user stories* relativas ao WP1, parte mais importante do projeto, foram atualizadas e completadas de forma a cobrir todas as suas necessidades. Foi também elaborado um estudo e registada a informação recolhida do cliente relativa às tarifas aplicadas no projeto *Green*. Esta informação foi sintetizada num documento com o objetivo de facilitar a percepção deste complexo requisito.
9	21-11-2011 a 27-11-2011	Nesta semana, iniciei um ficheiro com a especificação de *uses cases*. Este ficheiro está a ser realizado com o intuito de esclarecer algumas dúvidas no procedimento e execução de algumas tarefas. Para poder questionar o cliente sobre determinadas classes, estive a participar na realização de um ficheiro com a síntese das classes mais problemáticas.
10	28-11-2011 a 04-12-2011	Efetuei o levantamento e registo das dúvidas existentes, advindas da alteração de requisitos e de caraterísticas do projeto ainda não definidas claramente ou que não tivessem sido abordadas. Devo referir que estas dúvidas não provêm de todo o projeto, mas apenas do WP1, parte desenvolvida do projeto, e do WP3, parte seguinte a ser prototipada. Para melhorar a performance do projeto e obedecer a todos os requisitos do cliente realizei testes ao protótipo.
11	05-12-2011 a 11-12-2011	Nesta semana foram realizadas várias reuniões com o cliente. Nestas reuniões foram esclarecidas dúvidas e validado o protótipo, bem como, solicitadas alterações. No seguimento destas reuniões, realizei um documento com as alterações e retificações que teriam de ser efetuadas, nomeadamente a nível do protótipo e da base de dados.
12	12-12-2011 a 18-12-2011	Nesta semana, concluí a síntese da informação recolhida nas reuniões e esta, foi comunicada à equipa. Após este trabalho, introduzi as *user stories* no *Eventum* e comecei a realização de testes.
13	19-12-2011 a 25-12-2011	Realizei testes de aceitação ao projeto para verificar se todas as *user stories* planeadas estavam implementadas no programa do projeto *Green*. Registei erros, possibilidades de melhoria e *user stories* ainda não implementadas.
14	26-12-2011 a 01-01-2012	Interrupção do trabalho na *iUZ Technologies*.

Semana	Intervalo de dias	Descrição das tarefas
15	02-01-2012 a 08-01-2012	Efetuei a síntese do trabalho já efetuado no protótipo, tendo em conta os objetivos propostos (*user stories*) e os objetivos já alcançados. Realizei alterações nas *user stories* ainda não implementadas, tendo em conta as alterações já efetuadas nos requisitos.
16	09-01-2012 a 15-01-2012	Estudei o *feedback* do cliente relativamente à *release* de 22-12-2012. Realizei um documento com a síntese de todas as alterações e inseri novas *user stories* para desenvolver e *issues* de melhoria para corrigir pormenores que o cliente desejava de uma forma distinta do implementado. Iniciei a realização de uma arborescência[2] que pretendia ter todo o mapa do projeto *Green*. Este gráfico foi desenvolvido com o intuito de proporcionar à equipa de trabalho uma ideia clara e precisa das diversas funções que cada subdomain poderia realizar e, ainda mais concretamente, a sua precedência e dependência das diversas tarefas.
17	16-01-2012 a 22-01-2012	Nesta semana realizei um documento com as páginas que faltavam no protótipo, referentes ao WP3. Esse documento continha as funcionalidades que teriam de ser efetuadas em cada "página". À medida que o protótipo ia sendo desenvolvido, realizei alguns testes de desenvolvimento, para verificar se todas as funcionalidades eram realizáveis no novo protótipo.
18	23-01-2012 a 29-01-2012	Preparei a reunião com o cliente do Projeto *Green*. Realizei a participação na reunião com o cliente para esclarecimento de dúvidas e validação do protótipo. Realização de diagramas de estado e de sequência resultantes da reunião.
19	30-01-2012 a 05-02-2012	Continuação da realização de diagramas de estado e sequência. Atualização das *user stories* da próxima *sprint* face às informações recolhidas da reunião. Realização de testes de produção.

[2] Uma arborescência é uma árvore que possui uma raiz. Aplica-se a grafos orientados (Mariani, n. d.)

Semana	Intervalo de dias	Descrição das tarefas
20	06-02-2012 a 12-02-2012	Preparação de *user stories* para testes. Continuação de realização de testes na aplicação em testes de produção.
21	13-02-2012 a 19-02-2012	Esclarecimento de várias dúvidas advinda da equipa de trabalho com o *product owner*. Preparação das *user stories* para a realização de testes.
22	20-02-2012 a 26-02-2012	Realização de testes da aplicação em testes de produção e testes de desenvolvimento.
23	27-02-2012 a 04-03-2012	Completar as *user stories* necessárias para a *sprint* a decorrer. Verificação do protótipo relativo à *sprint* iniciada, se este continha todos os campos necessários e cumpria todos os requisitos. Realização de testes na aplicação em testes de produção.
24	05-03-2012 a 11-03-2012	Realização de testes na aplicação em testes de produção.
25	12-03-2012 a 18-03-2012	Realização de testes na aplicação em testes de produção. Preparação do guia para mostrar ao cliente a aplicação em testes de produção. Participação em reuniões com o cliente de aprovação da aplicação e validação do protótipo. Atualização e correção de *user stories* que seriam iniciadas na *sprint* seguinte.
26	19-03-2012 a 25-03-2012	Elaboração de procura A e B em testes de produção. Participação na atualização do algoritmo das tarifas e a verificação da atualização do protótipo. Continuação da realização de testes na aplicação em testes de produção.
27	26-03-2012 a 01-04-2012	Teste das tarifas em testes de desenvolvimento. Análise detalhada sobre o algoritmo das tarifas, os seus resultados, falhas e melhorias a implementar.
28	02-04-2012 a 08-04-2012	Atualização e retificação das *user stories* utilizadas na *sprint* seguinte. Análise e descrição de melhorias que poderiam ser implementadas no protótipo relativo às ofertas. Teste de tarifas em testes de produção.
29	09-04-2012 a 15-04-2012	Realização de testes em testes de produção sobre as tarifas, a possibilidade de aceder a diversos gestores de procura e portal. Análise do protótipo das ofertas e atualização do mesmo para cumprimento de todos os requisitos.

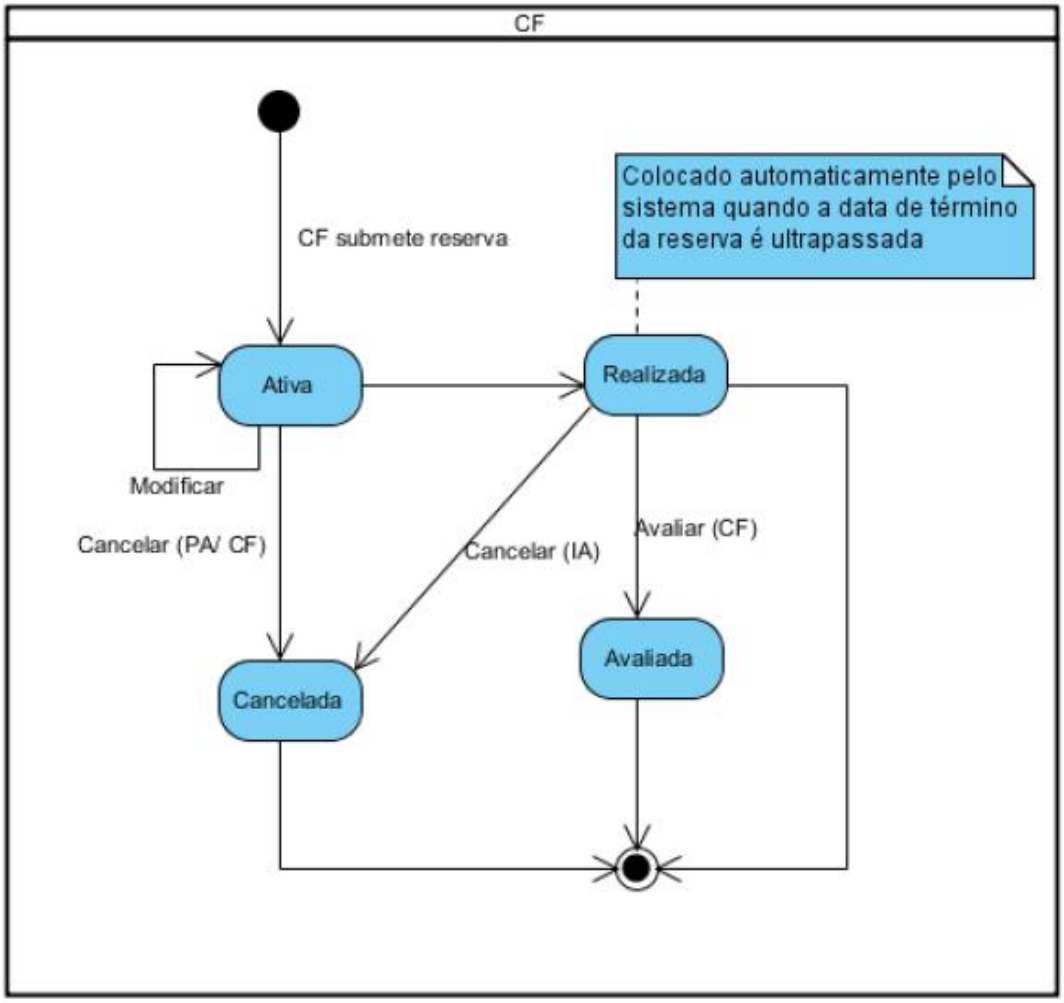
Semana	Intervalo de dias	Descrição das tarefas
30	16-04-2012 a 22-04-2012	Realização de novas tarefas de *scrum master*, tarefa realizada por mim na *sprint* 8 constituída na sua maioria por ofertas. Elaboração de um documento com os principais objetivos da *sprint*. Realização e direção de reuniões diárias nas quais se especificavam as tarefas realizadas no dia anterior, as tarefas que iriam ser elaboradas nesse dia e se descreviam impedimentos que poderiam existir para a realização do trabalho. Quando surgiam impedimentos, era traçada uma estratégia pela equipa de trabalho para que este pudesse ser eliminado e não fosse impeditivo para a realização do trabalho de um membro da equipa de desenvolvimento da aplicação. Atualização do quadro que continha as *user stories* desta *sprint* podendo estar no estado por realizar, em desenvolvimento, realizada e testada. Realização de testes de qualidade em testes de produção.
31	23-04-2012 a 29-04-2012	Continuação com as funções de *scrum master*. Realização de reuniões diárias. Atualização do quadro das *user stories*. Preparação de novas *user stories* de funcionalidades requeridas pelo cliente, consideradas de alta prioridade. Introdução dessas *user stories* na *sprint* a decorrer. Realização de testes de qualidade em testes de produção. Elaboração do balanço da *sprint* e ponto de situação das funcionalidades implementadas. Levantamento dos requisitos e funcionalidades necessárias para a atualização do protótipo dos serviços extra.
32	30-04-2012 a 06-05-2012	Realização de teste de qualidade em testes de produção. Preparação da reunião com o cliente em que elaborei um plano de testes para demonstrar todas as funcionalidades novas implementadas no sistema – ofertas, consulta de serviços e avaliação de serviços. Participação nas reuniões com o cliente em que se realizou a demonstração das novas funcionalidades implementadas na aplicação, o apontamento de sugestões de melhoria e correção de erros existente. Demonstração e análise do protótipo relativo às próximas funcionalidades a implementar.

Semana	Intervalo de dias	Descrição das tarefas
33	07-05-2012 a 13-05-2012	Realização de teste de aceitação em testes de produção no projeto *Green*.
34	14-05-2012 a 20-05-2012	Realização de teste de aceitação em testes de produção no projeto *Green* relativo a ofertas e tarifas. Elaboração e atualização das *user stories* para a próxima *sprint* que seria essencialmente de correção de *bugs*.
35	21-05-2012 a 25-05-2012	Levantamento de dúvidas existentes por parte da equipa de trabalho, para posterior esclarecimento. Continuação da realização de teste de aceitação em testes de produção no projeto *Green* relativo a ofertas e tarifas e inicio de testes em serviços extras. Análise do protótipo de contabilidade, funcionalidades a desenvolver após o término do estágio.

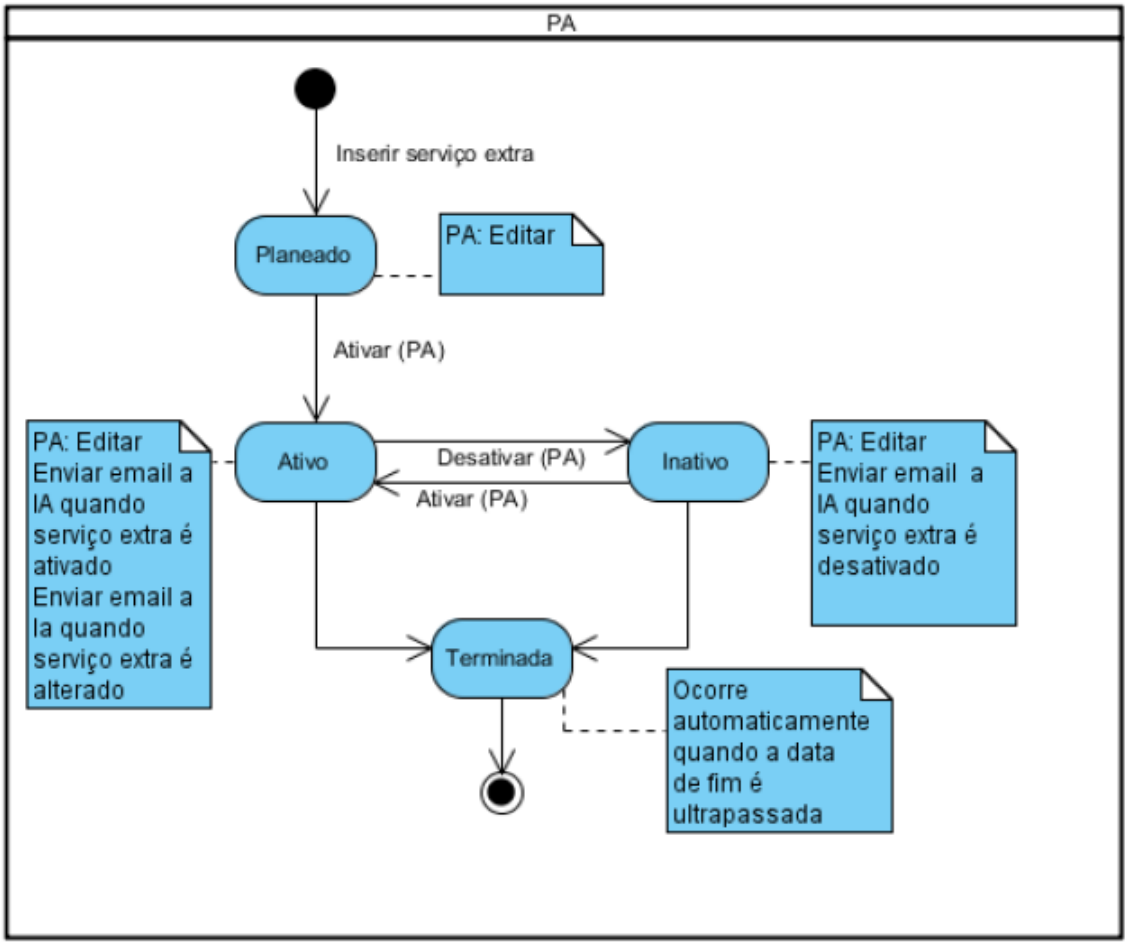
ANEXO 2

Ao longo da realização do projeto *Green* foram realizados vários diagramas de estado e de sequência para auxiliar a elaboração do projeto. Segue-se uma seleção de 3 diagramas de estado exemplificativos.

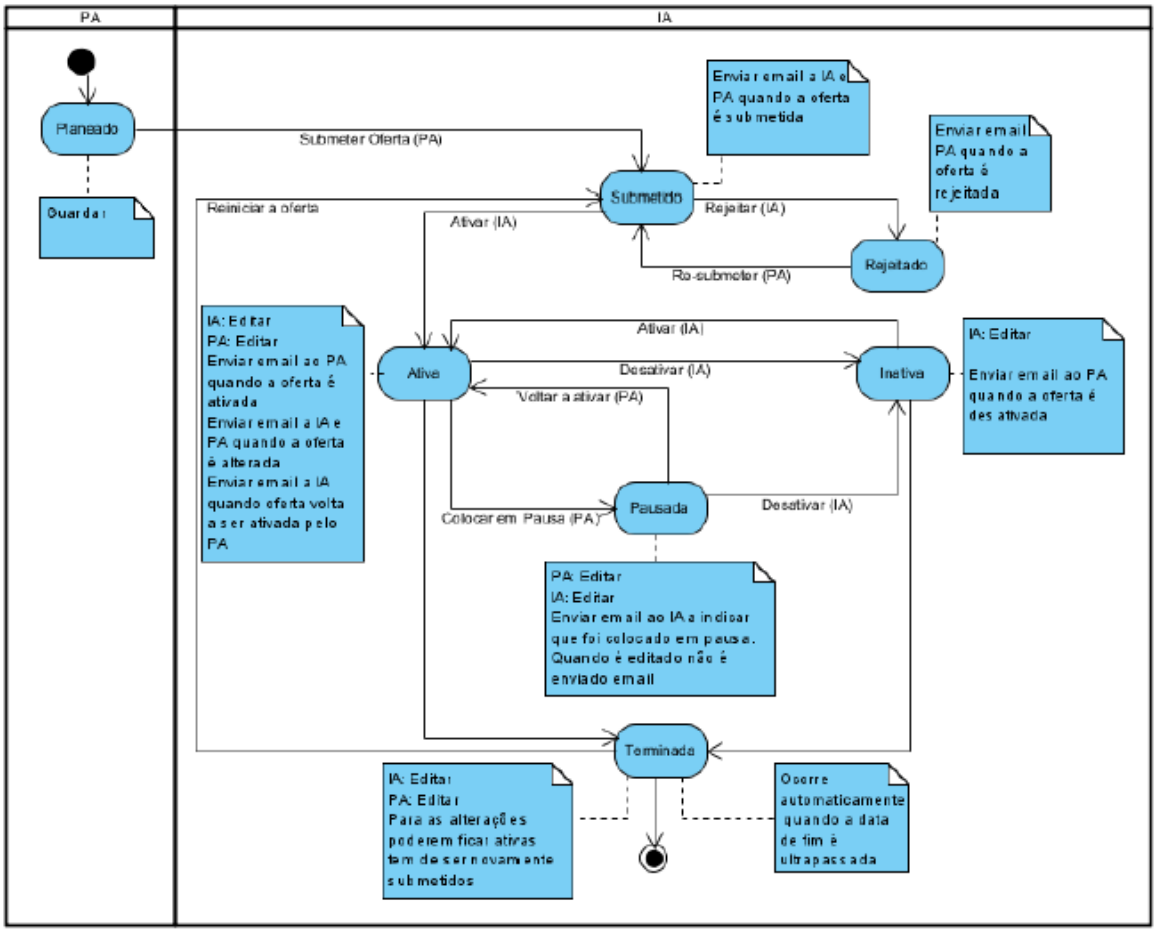
Estados dos serviços



Estados de serviços extra



Estados de ofertas



ANEXO 3

iUZ Technologies Lda.
Rua de Ceuta N.º 7
3800-009 Aveiro, Portugal
Tel.: +351 234 482 129 / Fax: +351 300 000 704
www.iuz.pt / contact@iuz.pt

DINA RAQUEL RODRIGUES RETROZ E SILVA

Carta de Recomendação Profissional

Ao longo dos últimos 8 (oito) meses, a iUZ Technologies teve o privilégio de contar com os contributos da Dina Retroz, como Consultora Funcional para a atividade de desenvolvimento de sistemas de informação.

Tratando-se de um período de formação (estágio curricular em contexto empresarial), importa realçar o nível de empenho, capacidade de aprendizagem e profissionalismo com que a Dina Retroz encarou e superou os desafios que lhe foram apresentados.

Salientamos de igual forma:

- **O impacto e importância, do trabalho da Dina Retroz, no processo produtivo da iUZ;**
- **Os contributos para a melhoria da capacidade e qualidade do trabalho realizado na iUZ;**

É por isso que, com elevada certeza, recomendamos e atestamos o potencial e competência da Dina Retroz, nas áreas da consultoria e verificação funcional de sistemas de informação, bem como no planeamento e organização de projetos de desenvolvimento de software.

No seguimento desta recomendação, reforçamos também a nossa proposta de continuidade de colaboração da Dina Retroz com a iUZ Technologies.

Aveiro, 13 de Junho de 2012

iUZ Technologies

iUZ Technologies Lda.
NIPC 507 976 460