SCHOOL OF ECONOMICS & MANAGEMENT LISBON

MESTRADO

GESTÃO DE SISTEMAS DE INFORMAÇÃO

TRABALHO FINAL DE MESTRADO

DISSERTAÇÃO

MÉTODOS DE TESTES APLICACIONAIS

SOFIA ISABEL CASTRO DE JESUS

09-2013

SCHOOL OF ECONOMICS & MANAGEMENT LISBON

MESTRADO EM

GESTÃO DE SISTEMAS D INFORMAÇÃO

TRABALHO FINAL DE MESTRADO

DISSERTAÇÃO

MÉTODOS DE TESTES APLICACIONAIS

SOFIA ISABEL CASTRO DE JESUS

ORIENTAÇÃO:

PROFESSOR JESUALDO FERNANDES

09-2013

Agradecimentos

Uma dissertação é um trabalho individual, mas para mim esta dissertação foi realizada graças à orientação e apoio da minha família, amigos e colegas de trabalho. Quero agradecer especialmente:

…ao meu orientador Jesualdo Fernandes pela disponibilidade e orientação.

…aos meus pais que sempre apoiaram as minhas escolhas e proporcionaram esta oportunidade.

…ao António Gomes, Paulo Ascenção e Sara Lopes que tornaram possível este projeto.

…ao André Santos e André Lopes pela preocupação, apoio e amizade demonstrados.

…ao Lourenço Maia pela paciência e carinho.

Índice

Índice de Figuras

Índice de Quadros

Resumo

A atividade de Testes Aplicacionais é uma componente integrante do ciclo de desenvolvimento de uma aplicação. Inicialmente, esta tarefa estava intrinsecamente ligada à própria atividade de programação, não existindo uma definição propriamente dita. A sua evolução passou pelo desenvolvimento de técnicas, ferramentas e processos próprios que ainda hoje são alvo de melhoria.

Nesta dissertação foi estudada a evolução dos Testes Aplicacionais, o seu conceito, objetivos e tipos. Pretende-se demonstrar a importância dos Testes Aplicacionais, nomeadamente, dos Testes Funcionais e quais são os critérios de escolha de um método de Testes Funcionais adequado a um determinado negócio.

Das pesquisas efetuadas foi elaborado uma revisão da literatura e para complementar foi efetuado um Estudo de Caso numa instituição bancária onde foi estudado o seu método de execução de Testes Funcionais.

O principal resultado obtido foi a definição dos critérios de seleção de um método de Testes Funcionais. Esses critérios passam pela cultura da empresa, boas práticas, a tecnologia adotada, os custos e os próprios modelos de desenvolvimento de *software*.

Palavras-Chave: Testes Aplicacionais, Métodos, Testes Funcionais.

Abstract

Applicational Testing activity is an important component of the application development cycle. Initially this activity was intrinsically linked to the programming activity, but there wasn't a specific definition. Its evolution passed through the development of techniques, tools and processes that today still are being improved.

In this dissertation it was studied the Applicational Testing evolution, its concept, objectives and types. The focus of this study lies on Functional Tests, which are a type of Applicational Testing, and the respective methods. We intend to demonstrate the importance of Applicational Testing, mainly the Functional Testing and which are the criteria for choosing the appropriate method of Functional Tests that suits best for a particular business.

For this research, a literature review was carried out, as well, in a banking institution a case study was elaborated in which their method of performing Functional Testing was studied.

The main result that was bought about was the selection criteria for the functional testing method. Those criteria are: organizational culture, well-practice, the adopted technology, costs and the own models for *software* development.

Key-words: Applicational Tests, Methods, Functional Tests.

Glossário

Área de Processo: conjunto de práticas relacionadas que, quando implementadas coletivamente, satisfazem um conjunto de metas consideradas importantes para a melhoria nessa área.

Caso de Teste: conjunto de *inputs* de teste, condições de execução e resultados esperados, desenvolvido para atingir um determinado objetivo.

Core business: negócio/atividade principal de uma empresa.

Debugging: Ato de *debug*, que significa detetar, localizar e corrigir erros em programas de computador.

Homebanking: serviço disponibilizado pelos bancos que permite aos clientes registados efetuar vários tipos de operações bancárias através de um equipamento ligado à Internet.

Outsourcer: empresa que fornece o serviço de *outsourcing.*

Outsourcing: contratação externa de serviços que, originalmente, eram realizados por colaboradores internos, com o fim de reduzir os custos e melhorar os serviços; subcontratação.

Tecnologias de Informação: conjunto de recursos tecnológicos e computacionais para criação, aplicação e uso da informação.

Abreviaturas

CMM – Capability Maturity Model

CMMI – Capability Maturity Model Integration

ER – Especificação de Requisitos

ISTQB - International Software Testing Qualifications Board

ITIL - Information Technology Infrastructure Library

ISO - International Organization for Standardization

NIB – Número de Identificação Bancária

PdS – Ponto de Situação

SEI – Software Engineering Institute

XP – Extreme Programming

1. Introdução

Esta dissertação enquadra-se no âmbito do Mestrado em Gestão de Sistemas de Informação e o tema abordado são os Testes Aplicacionais e respetivos métodos. Foi estudada, a evolução, conceito, tipos e técnicas de teste, bem como os seus métodos associados. Como metodologia, optou-se pela realização de um Estudo de Caso numa instituição bancária nacional. A escolha do tema em questão advém do interesse pessoal e profissional da autora. O Estudo de Caso foi selecionado como metodologia devido à natureza da questão de investigação, devido a não ser necessário observar eventos comportamentais e devido ao tema ser atual.

Quando é efetuada uma transferência, através de qualquer canal (*Homebanking,* caixas de multibanco, etc.) existem expectativas que necessitam ser satisfeitas, como por exemplo, saber que o dinheiro vai ser transferido para o destinatário pretendido. Para que os clientes possuam essa confiança é necessário que a aplicação seja fiável. Para isso, são efetuadas várias transferências "fictícias" de forma a verificar o comportamento da aplicação e corrigir eventuais erros existentes antes de disponibilizar a aplicação para o cliente. Essa atividade consiste em executar Testes Aplicacionais.

Os Testes Aplicacionais assumiram-se gradualmente como uma componente de elevada importância no ciclo de vida de um produto, devido ao aumento de qualidade proporcionado nas aplicações e consequente redução de custos com manutenções futuras (Gelperin & Hetzel 1988).

Inicialmente, não existia um método para executar Testes Aplicacionais, ou seja, os procedimentos eram efetuados numa perspetiva *ad-hoc* o que não era muito eficiente. Para colmatar este facto surgiu a necessidade de criar métodos e *standards* para os Testes Aplicacionais (Gelperin & Hetzel 1988).

No âmbito do tema desta dissertação define-se a atividade "Testar" como a execução de um programa através da introdução de dados artificiais, de forma a obter resultados que permitam descobrir erros, anomalias ou informação acerca dos atributos não funcionais do programa (Myers 2004). Testar é entendido como o ato de efetuar Testes Aplicacionais.

Para efeitos de delimitação da dissertação, o estudo incide sobre os Testes Funcionais, contudo foi elaborada uma contextualização no âmbito dos Testes Aplicacionais. Os Testes Funcionais são um tipo de Testes Aplicacionais que têm como propósito testar estrutural e funcionalmente as aplicações como um todo de forma a verificar inconsistências em relação às especificações das aplicações (Sommerville, 2011; Li, 1990 e IEEE Standards Board, 1990).

O objetivo do estudo consiste em responder à seguinte questão de investigação: "Como devem as empresas escolher o método de Testes Aplicacionais mais adequado ao seu negócio?". No seguimento desta questão surgem outras que merecem ser exploradas, tais como: "É possível aplicar diferentes métodos de Testes Aplicacionais a diferentes aplicações?" e "Como se devem avaliar os resultados obtidos da implementação de um determinado método?".

A estrutura deste estudo divide-se numa componente teórica e noutra empírica. A componente teórica tem como objetivo exibir alguns estudos já elaborados sobre o tema em causa, apresentando os tópicos considerados relevantes. Devido a limitações de tamanho e tempo existem alguns aspetos que não foram abordados, tais como os riscos dos Testes Aplicacionais, eventuais problemas que surgem no decorrer na execução do método escolhido, entre outros.

A componente empírica consiste num Estudo de Caso numa instituição bancária nacional, onde se estudou o método de Testes Funcionais aplicado. As técnicas de recolha de dados basearam-se em entrevistas, observação direta e o estudo de alguns documentos.

2. Revisão da Literatura

2.1. Testes Aplicacionais

2.1.1. Origem e evolução dos Testes Aplicacionais

A realização de testes é tão antiga como a programação, contudo os seus objetivos, ferramentas, técnicas e métodos têm vindo a evoluir ao longo do tempo. O seu conceito tem ganho contornos cada vez mais nítidos e a sua importância tem subido no *ranking* de prioridades das empresas (Gelperin & Hetzel 1988).

Inicialmente, apenas existiam testes a *Hardware* e estes eram feitos à medida que o código era escrito. Testar não era considerada uma tarefa/atividade isolada, estando interligada com o ato de programar (Gelperin & Hetzel 1988).

Gelperin e Hetzel (1988) descreveram a evolução dos Testes Aplicacionais até então, dividindo as várias fases ao longo do tempo da seguinte forma:

Período	Fase
Até 1959	*Debugging-Oriented Period*
1957-1978	*Demonstration-Oriented Period*
1979-1982	*Destruction-Oriented Period*
1983-1987	*Evaluation-Oriented Period*
1988-actualmente	*Prevention-Oriented Period*

Quadro 1- Evolução dos Testes Aplicacionais
Adaptado de Gelperin & Hetzel (1988)

Na década de 50, Alan Turing escreveu os primeiros artigos sobre o tema, onde lançou uma questão pertinente que "despertou" o grande objetivo ou a grande razão de ser dos Testes Aplicacionais. A questão discutida foi: "Como sabemos que um programa é/exibe inteligência?". Esta questão traduzida para um caso real, definiu-se da seguinte forma: "Como sabemos que um programa satisfaz os seus requisitos?" (Gelperin & Hetzel 1988, p.689).

Até existir uma visão mais clara sobre esta temática, "testar" e "*debugging"* eram conceitos indistinguíveis. Segundo IEEE Standards Board (1990) *debugging é* o ato de *debug*, que significa detetar, localizar e corrigir erros num programa de computador.

Gelperin & Hetzel (1988) referiram que na fase *Debugging-Oriented* já se diferenciavam os conceitos de "testar" e "*debugging",* mas as suas diferenças eram difíceis de explicar, tendo este problema de definição perdurado durante algum tempo sem que se conseguisse, de certa forma, fazer uma diferenciação clara entre os dois termos.

Antes da resolução deste problema, a definição de Testes Aplicacionais começou a ganhar contornos devido às razões enumeradas de seguida, as quais são também justificativas da situação atual, em termos de evolução dos Testes Aplicacionais (Gelperin & Hetzel 1988).

As principais razões pelas quais os Testes Aplicacionais ganharam um maior protagonismo são as seguintes (Gelperin & Hetzel 1988):

- Aumento das aplicações em termos de número, custo e complexidade;
- Aumento das alterações nas aplicações existentes;

- Necessidade de diminuição do número de falhas nas aplicações e consequentemente aumento da sua eficiência – constatou-se que o custo de reparar erros depois da aplicação estar em produção é maior do que investir em Testes Aplicacionais. Entenda-se por "colocar/entrar em produção" como a disponibilização das aplicações ao utilizador final;
- Crescente preocupação com a qualidade das aplicações.

Na fase *Demonstration-Oriented* testar envolvia dois objetivos principais: "Make sure the program runs" e "Make sure the program solves the problem" (Gelperin & Hetzel 1988, p.689). Os testes aplicacionais tinham como pretensão demonstrar correção, onde um teste ideal seria bem sucedido quando o resultado obtido não apresentava erros (Gelperin & Hetzel 1988).

Por volta da década de 80, surgiu a fase *Destruction-Oriented,* cujo objetivo dos Testes Aplicacionais passou a ser exatamente o contrário do objetivo descrito na fase anterior. Enquanto, anteriormente, um teste bem sucedido era aquele onde não se encontravam erros, na fase *Destruction-Oriented*, tal como o nome indica, o principal objetivo era encontrar erros nas aplicações. Segundo Myers (2004, p.11) testar é definido com o "processo de executar um programa com a intenção de encontrar erros".

Foi entre a fase *Demonstraion-Oriented* e a fase *Destruction-Oriented* que foram distinguidos os conceitos "testar" e "*debuging"* (Quadro 2). Enquanto na primeira os dois conceitos tinham a mesma definição, na fase *Destruction-Oriented* houve uma distinção mais clara, onde "testar" detetava erros e "*debuging*" localizava-os, identificava-os e corrigia-os.

	Destruction Model	**Demonstration model**
Test	Detect	Detect Locate Identify Correct
Debug	Locate Identify Correct	Detect Locate Identify Correct
Objetivos	Test: Detect faults *Debug*: Fix faults	Test=*Debug* Make sure it runs Make sure it solves de problem

Quadro 2 - Diferença entre Testar e *Debug*
Adaptado de Gelperin & Hetzel (1988)

Na fase *Evaluation-Oriented* os testes aplicacionais já estavam presentes nas várias etapas do ciclo de vida do produto e já tinham um peso grande para a obtenção de sucesso das aplicações. Em 1983, o Instituto de Ciência Computacional e de Tecnologia (*Institute for Computer Sciences and Technology of the National Bureau of Standards*) publicou um guia de validação, verificação e teste durante o ciclo de vida das aplicações (*Guideline for Lifecycle Validation, Verification and Testing of Computer Software*), onde foi descrita uma metodologia que integra as atividades de análise, revisão e testes de forma a proporcionar uma avaliação do produto ao longo do seu ciclo de vida. Nesta fase a importância da qualidade das aplicações começou a crescer, bem como a necessidade de diminuir os custos com a manutenção e correção de erros depois da implementação das aplicações. Como o próprio nome indica, o objetivo principal residia na avaliação constante do produto à medida que este ia sendo desenvolvido (Gelperin & Hetzel 1988).

Na última fase estudada pelos autores Gelperin & Hetzel (1988) (*Prevention-Oriented*) surgiu a metodologia STEP (*Systematic Test and Evaluation Process*), baseada num modelo de prevenção no ciclo de vida do produto, que encara a tarefa de testar de forma paralela à tarefa de desenvolvimento, com uma sequência de atividades incluindo

planeamento, análise, desenho, implementação, execução e manutenção (Gelperin & Hetzel 1988).

Tanto na fase *Evaluation-Oriented* como na *Prevention-Oriented* a tarefa de desenho de Casos de Teste é bastante relevante para o alcance dos objetivos de testes, sendo Casos de Teste um conjunto de *inputs*, condições de execução e resultados esperados desenvolvidos para verificar o cumprimento da Especificação de Requisitos (IEEE Computer Society 2005). Especificação de Requisitos é um documento onde estão redigidas as condições (requisitos) de uma aplicação (IEEE Standards Board 1990).

Nesta última fase, em específico, as atividades de planeamento, análise e desenho de Casos de Testes melhoram a qualidade da Especificação de Requisitos das aplicações, no sentido em que nestas atividades são levantadas questões que revelam lacunas, ambiguidades, inconsistências e incorreções na Especificação de Requisitos (Gelperin & Hetzel 1988). Ao identificar estas situações durante as primeiras atividades, é mais fácil, rápido e barato adaptar as aplicações às alterações necessárias, evitando posteriores erros.

O desenho de Casos de Teste juntamente com a elaboração de modelos permite aos *testers* (quem testa) obter uma melhor compreensão do comportamento das aplicações dado determinados *inputs*, sendo desta forma, possível alertar para mais situações de lacuna, inconsistência, etc. (Gelperin & Hetzel 1988).

2.1.2. Conceito e objetivos dos Testes Aplicacionais

Atualmente, existe um processo de Verificação e Validação que pretende aumentar a qualidade das aplicações ao longo do seu ciclo de vida. Deste processo fazem parte, entre outros elementos, os Testes Aplicacionais (IEEE Computer Society 2005).

Verificação consiste no conjunto de atividades que assegura a implementação correta de uma função específica. Validação refere-se a um conjunto diferente de atividades que pretende assegurar que é possível rastrear os requisitos nas aplicações desenvolvidas (Sommerville 2011). Sommerville (2011, P.208) cita Boehm ao definir verificação com a questão "Estamos a construir o produto bem?" e validação com a questão "Estamos a construir o produto certo?".

O processo de Verificação e Validação inclui tarefas como revisões formais técnicas, auditorias de qualidade e configuração, monitorização de *performance*, simulação, estudo de viabilidade, revisão de documentação e base de dados, análise de algoritmos e realização de testes aplicacionais (Pressman 2001).

O conceito e os objetivos dos testes aplicacionais sofreram alterações ao longo do tempo como foi mencionado anteriormente. Posto isto, existem várias definições segundo diferentes autores (Gelperin & Hetzel 1988; Sommerville 2011; Myers 2004), que, embora tenham surgido em épocas diferentes, demonstram as várias perspetivas relevantes do que significa testar aplicações.

Foram identificadas 3 tipos de perspetivas relativamente ao significado da tarefa Testar.

Segundo Gelperin & Hetzel (1988), como já foi visto, testar tinha como principal objetivo a demonstração de correção, onde um teste ideal seria bem sucedido quando o programa não continha erros.

Segundo Myers (2004) testar é o processo de executar um programa com o principal objetivo de encontrar erros. Para além disso, testar tem também como propósito acrescentar valor à aplicação, aumentando a sua qualidade. Um teste bem sucedido é aquele onde foi possível encontrar algum erro que, depois de corrigido, irá melhorar a *performance* da aplicação. Myers (2004) definiu alguns princípios para os Testes

Aplicacionais que devem ser seguidos de forma a atingir as metas pretendidas (Quadro 3).

Finalmente, a última perspetiva de indicação relevante é a de Sommerville (2011). Este autor defende que testar consiste na execução de um programa com dados artificiais, de forma a descobrir erros, anomalias ou informação acerca dos atributos não funcionais, através dos resultados obtidos. Nesta perspetiva, testar tem como propósitos: mostrar que um programa faz o que é suposto fazer e descobrir erros antes da entrada em produção.

A primeira perspetiva é oposta às duas seguintes, pois, enquanto a primeira (Gelperin & Hetzel 1988) considera testar um processo destrutivo (pelo menos psicologicamente), onde se pretende encontrar erros; a segunda (Myers 2004), por sua vez, considera testar um processo de mostrar que o programa não contém erros. Neste caso, é claro que existe aqui uma evolução e, como já foi explicado anteriormente, a perspetiva de Gelperin & Hetzel (1988) não trazia benefícios para o produto, no entanto é importante notar que testar não foi algo fácil de definir e ainda hoje existem diferentes visões.

Número	Princípio
1	Uma parte essencial de um Caso de Teste é a definição do resultado esperado.
2	Um programador deve evitar testar a aplicação que desenvolveu.
3	Uma empresa não deve testar as próprias aplicações.
4	Inspecionar cuidadosamente o resultado de cada teste.
5	Os Casos de Teste devem ser escritos possibilitando, tanto o teste com dados válidos como inválidos.
6	Examinar a aplicação de forma a perceber se esta não faz o que não é suposto fazer é apenas metade da batalha; a outra metade é perceber se o programa faz o que não é suposto fazer.
7	Evitar eliminar Casos de Teste, a não ser que a aplicação também seja eliminada.
8	Não planear o esforço de execução de testes sob a assunção de que não serão encontrados erros.
9	A probabilidade da existência de mais erros numa determinada componente ou funcionalidade é proporcionalmente igual ao número de erros já encontrados nessa componente ou funcionalidade.
10	Testar é uma tarefa extremamente criativa e desafiante intelectualmente.

Quadro 3 – Principais princípios da execução de Testes Aplicacionais
Adaptado de Myers (2011)

2.1.3. Tipos de testes

Este trabalho irá focar-se nos Testes Funcionais. Contudo existem outros tipos de testes, como podemos observar no seguinte quadro (Sommerville 2011; Li 1990; IEEE Standards Board 1990):

Tipo de Teste	Descrição
Teste de Componentes ou Unitário	Processo de testar individualmente subprogramas, subrotinas ou procedimentos numa aplicação, com diferentes parâmetros, de forma a perceber se existem contradições em relação às especificações da aplicação.
Testes de Integração ou de Interface	Processo de testar os módulos em conjunto de forma a verificar se se interligam corretamente identificando incompatibilidades entre a interface dos vários módulos/componentes.
Testes de Sistema ou Funcionais	Processo de testar estrutural e funcionalmente as aplicações como um todo de forma a verificar inconsistências em relação às especificações das aplicações.
Testes de Regressão	Repetição de testes sobre um componente ou sobre a aplicação completa, após a realização de alterações, de forma a verificar que o componente ou a aplicação continua conforma a ER.
Testes de Aceitação	Testes efetuados pelo utilizador final, para que este possa verificar eventuais discordâncias em relação ao que foi pedido.
Testes de Instalação	Processo de testar a instalação da aplicação de forma a verificar se todas as componentes estão instaladas e se se interligam entre si.

Quadro 4 – Tipos de Testes Aplicacionais
Adaptado de Sommerville (2011); Li (1990) e IEEE Standards Board (1990)

Normalmente, os tipos de teste mencionados no quadro anterior são realizados pela ordem apresentada e alguns deles podem ser efetuados durante o desenvolvimento da aplicação, tal como mostra a figura 1 (ESA Board for Software Standardisation and Control 1995). O desenho detalhado da aplicação é validade nos Testes Unitários, a arquitetura da aplicação é validada nos Testes de Integração e os requisitos são validados nos Testes de Aceitação e nos Testes Funcionais. Para além destes, podem

ainda ser realizados testes de *performance*, de carga, de stress, entre outros, contudo os enunciados no quadro 4 são os mais usuais.

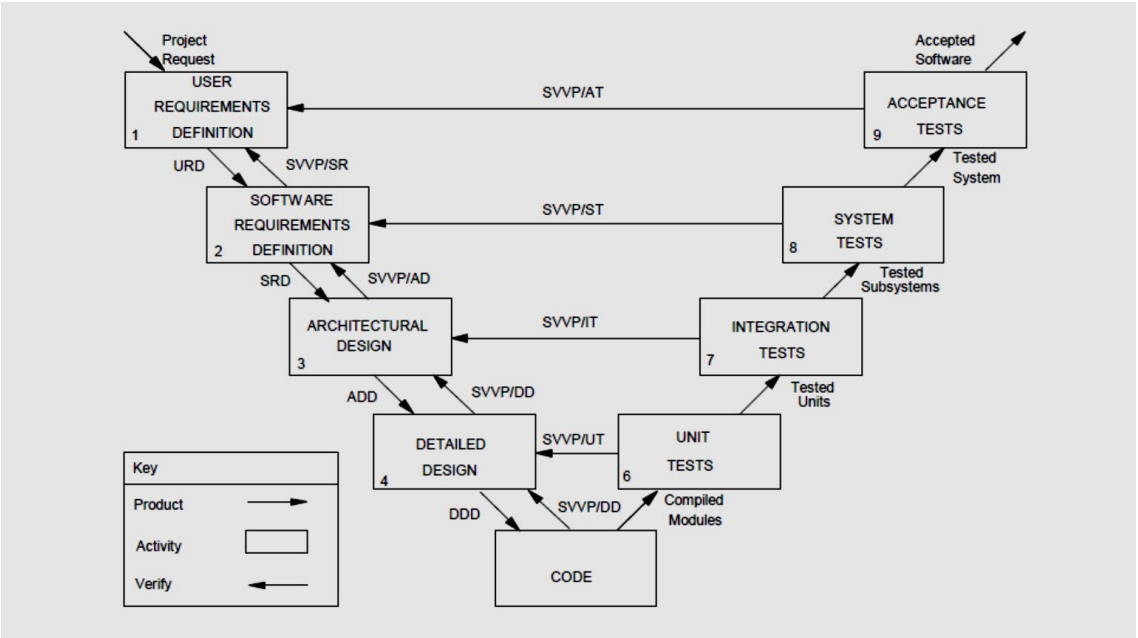

Figura 1 – Modelo em V

Fonte: ESA Board for Software Standardisation and Control (1995)

2.1.4. Vantagens e desvantagens da realização de testes através de outsourcing

Os Testes Aplicacionais podem ser realizados tanto por uma equipa interna ou por uma equipa externa, ou seja, através de *outsourcing*. Uma vez que no estudo empírico os testes funcionais são realizados maioritariamente por uma equipa através de *outsourcing*, é importante salientar as suas vantagens e desvantagens e perceber o seu conceito.

Entenda-se por *outsourcing* a contratação externa de uma entidade para gerir um processo de negócio de forma mais eficiente e eficaz (Aalders 2002).

"As organizações têm sempre contratado empresas para tipos de trabalhos particulares ou para obterem ajuda quando existem picos no seu volume de trabalho, formando

relações de longa duração com essas empresas cujas capacidades complementam ou suplementam as suas" (Alexandre & Cunha 2008, p.3).

No seguinte quadro são apresentadas as principais vantagens e riscos do *outsourcing*:

Vantagens	**Riscos**
Focalização na competência principal da empresa – por exemplo, a área de Tecnologias de Informação não é o *core business* da maioria das empresas.	Possibilidade de o subcontratado se revelar mais ineficaz e ineficiente do que o subcontratante.
Qualidade de serviço – o *outsourcer*, como prestador de serviços irá focar-se na melhoria de qualidade do seu serviço de forma a mantê-lo.	Inexperiência do subcontratado.
Pessoal de Qualidade e Especializado – ao contratar um serviço de *outsourcing*, contratam-se recursos especializados naquele serviço.	Incerteza quanto à evolução do negócio.
Flexibilidade no número de operários – é possível contratar um serviço de *outsourcing* por um montante fixo onde o número de recursos varia consoante as necessidades do *outsourcer* sem que o cliente se tenha de preocupar com os custos desses recursos.	Risco de surgirem subcontratados mais eficientes e com maior diversidade de soluções.
Melhor controlo da gestão de custos – mediante o tipo de contrato os custos de *outsourcing* são menos variáveis do que contratar recursos internamente para efetuar o mesmo trabalho.	Risco de surgirem melhores alternativas em relação ao desempenho das atividades.
Melhoria nas qualidades de gestão – o *outsourcer* tem objectivos rígidos a atingir o que implica um maior foco na gestão.	Risco de a organização se encontrar numa situação fragilizada e sem competências internas para se adaptar à mudança.
Manter competitividade com a competição – o cliente foca-se mais no seu *core business* e obtém maior eficiência e eficácia nas áreas em que estão em outsourcing.	Perigo de a subcontratação se traduzir no aparecimento de mais um interlocutor a dificultar a comunicação e a gerar conflitos entre as várias partes envolvidas.
Processos de negócio melhorados – os *outsourcer* incentivam o desenvolvimento do orçamento gerido pelo negócio e os gastos do projeto controlados pelo negócio.	Eventual ocorrência de custos ocultos.
O *outsourcer* pode ser despedido - o *outsourcer* poder ser despedido, enquanto que, por exemplo, não se pode despedir todo um Departamento de Tecnologias de Informação.	Tendência em considerar o *outsourcing* como um fim e uma solução definitiva, em vez de um meio de concentração de recursos em áreas vitais.

Quadro 5 – Vantagens e Riscos do *outsourcing*
Adaptado de Alexandre & Cunha (2008)

2.2. Método de testes

Até aqui, foi efetuada uma contextualização sobre os testes aplicacionais. O restante estudo irá incidir sobre o tema em questão, que são os testes funcionais. Para o âmbito em causa, apenas foi encontrada uma metodologia específica para os testes funcionais. Não obstante isso, existem também técnicas e ferramentas úteis para os Testes Funcionais.

Um método consiste na descrição das características de um processo ou conjunto de procedimentos utilizados num dado serviço ou para desenvolver um produto (IEEE Standards Board 1990); É uma estratégia; modo de proceder; esforço para atingir um fim (Porto Editora 2013).

2.2.1. Processo/Métodos de Testes Aplicacionais

Testar, como vimos na figura 1, acompanha todo o ciclo de vida de um produto de aplicações. No entanto, consoante o modelo de desenvolvimento, esta tarefa encaixa-se de forma diferente e nem todos os tipos de teste são efetuados em todos os produtos de aplicações.

Para ilustrar esta situação vamos observar alguns dos principais modelos de desenvolvimento.

Modelo sequencial linear (*Waterfall)* (Pressman 2001)

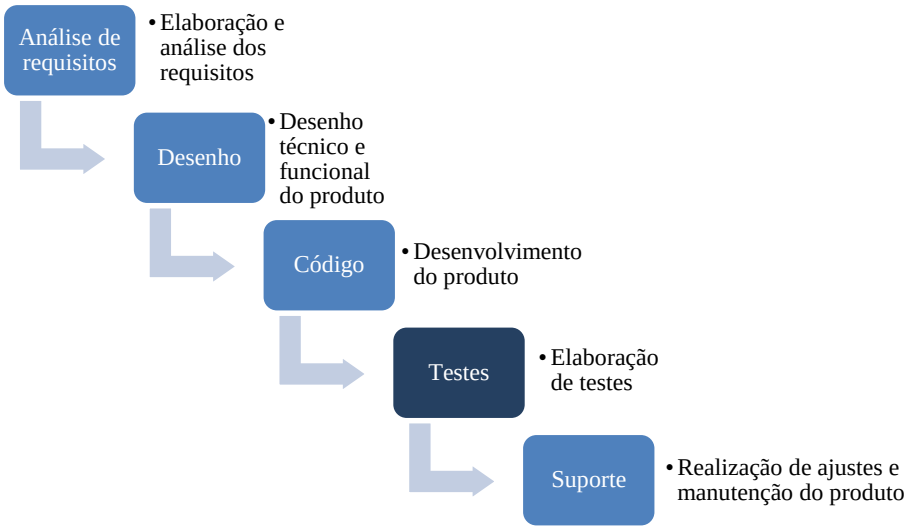

Figura 2 – Modelo sequencial linear
Adaptado de Pressman (2001)

Neste modelo existe uma sequência de fases e a passagem de uma para a outra acontece quando a primeira está finalizada (Pressman 2001).

Modelo de protótipo (Pressman 2001)

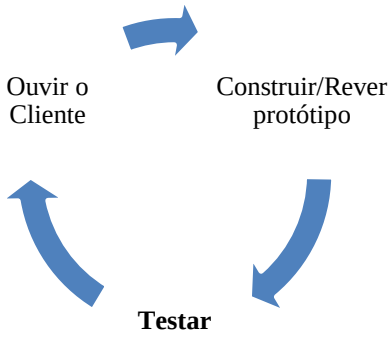

Figura 3 – Modelo de Protótipo
Adaptado de (Pressman 2001)

Este modelo, ao contrário do anterior é cíclico e o produto não é desenvolvido de uma só vez, existindo melhorias constantes em cada ciclo. Os testes são feitos à medida que são desenvolvidos/revistos protótipos (Pressman 2001).

Extreme Programming

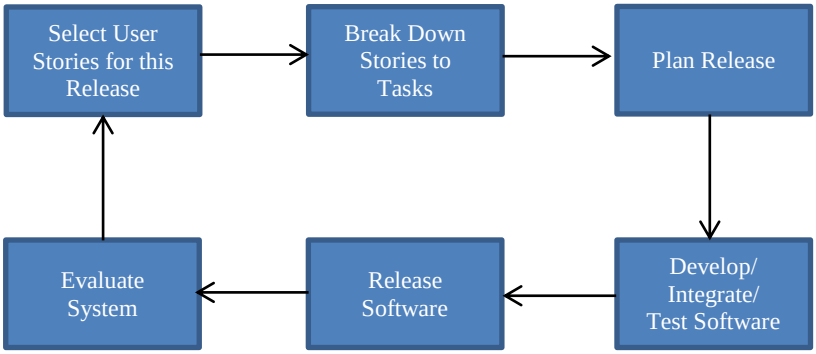

Figura 4 – Processo do *Extreme Programming*
Adaptado de Sommerville (2011)

Extreme Programming é um modelo que tem como base a modelação ágil. Este modelo consiste num conjunto de práticas sustentadas por princípios e valores com o propósito de fornecer uma forma de desenvolver uma aplicação com base em iterações. Entenda-se por modelar a realização de uma representação da aplicação, descrevendo as relações entre os vários objetos e tarefas (Scott Ambler 2002).

O processo de desenvolvimento de uma aplicação a partir do modelo XP é realizado através de ciclos/iterações. Em cada ciclo são realizadas as tarefas descritas na figura 4, com o objetivo de melhorar a qualidade da aplicação (Scott Ambler 2002). Em cada iteração existem *user stories* (cenários) que são decompostos em tarefas. Seguidamente, antes de escrever o código, são desenhados os testes a realizar após a programação e por último é feita uma avaliação ao sistema do que foi implementado na iteração (Sommerville 2011). É importante referir que o ciclo descrito na figura 4 por Sommerville não vai totalmente de encontro à perspetiva de outros autores tais como Beck e Scott Ambler.

Um dos princípios do XP que é de grande importância mencionar é o *Test Driven Development*. Este princípio traduz-se no desenho de testes antes de escrever o código. Tal como demonstra a figura 4, os testes são desenhados antes da programação

propriamente dita e enquanto forem detetados erros nos testes a iteração não pode ser dada como concluída. Outra particularidade deste princípio prende-se com a automatização dos testes, que facilita a realização de testes de regressão (Beck 1999).

Embora testar seja uma atividade que possa ser "encaixada" de formas diferentes em cada modelo, existe um processo/método principal para a executar referida por Sommerville (2011), cuja descrição podemos observar na figura 5:

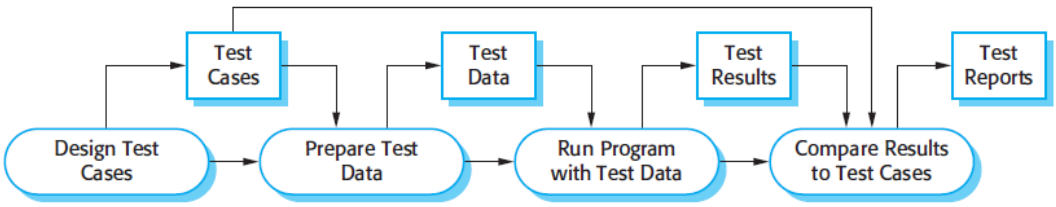

Figura 5 – Método de Testes Funcionais
Adaptado de Sommerville (2011)

Os Casos de Teste, bem como os dados necessários para efetuar os Casos de Teste, podem ser gerados automaticamente ou não, sendo que, como a geração manual destes é uma tarefa morosa, existem aplicações que o fazem automaticamente. A geração automática permite aumentar a rapidez do desenho de Casos de Teste, permite elaborar Casos de Teste mais complexos (como a simulação de centenas de utilizadores ligados à aplicação ao mesmo tempo), permite efetuar testes de regressão mais amplos, ou seja verificar todos os Casos de Teste depois de efetuada uma alteração e permite, também, aumentar a cobertura de testes efetuados à aplicação (R. Jeevarathinam 2010; Bernardo et al. 2008).

A principal forma de automatização tem como base a Especificação de Requisitos, mas também pode ter como base o código fonte da aplicação ou um conjunto de erros-tipo identificados noutras aplicações. Para os testes funcionais a automatização, normalmente, tem como base a Especificação de Requisitos, no entanto, é difícil

automatizar este tipo de testes devido à sua complexidade e natureza do próprio teste. A automatização de testes unitários está mais aprofundada, uma vez que a base é, por norma, o código fonte (Barbosa et al. n.d.). Porém, a automatização de testes também está intrinsecamente associada ao modelo de desenvolvimento. Como vimos, anteriormente, os testes, em XP, são, maioritariamente automatizados, qualquer que seja a sua tipologia (funcionais, unitários, etc.) (Beck 1999).

O próximo passo no processo consiste na execução dos Casos de Teste, ou seja, são introduzidos os dados definidos como *input* e são seguidos os passos descritos nos Casos de Teste, de forma a observar o resultado gerado pela aplicação. Esses resultados serão comparados com os resultados esperados dos Casos de Teste. Desta comparação podem resultar ou não erros na aplicação que devem ser reportados e resolvidos. No fim do processo são produzidos relatórios com os resultados obtidos (Sommerville 2011).

Existem várias ferramentas de suporte ao processo de execução de testes. Algumas ferramentas suportam o processo inteiro descrito na figura 5, outras apenas suportam parte, tal como a geração de Casos de Teste ou o registo dos erros encontrados durante a execução de testes. Estas ferramentas são usadas com o principal objetivo de melhorar a eficiência do processo de execução de testes, através da automatização de tarefas (Foundation Certificate in Software Testing 2010).

Para além das ferramentas de suporte existem, ainda, modelos para certificar as empresas em relação a *standards* de qualidade aplicacional. Alguns desses modelos são reconhecidos mundialmente como CMMI (*Capability Maturity Model Integration*), ITIL (*Information Technology Infrastructure Library*), ISTQB (*International Software Testing Qualifications Board*), certificações ISO (I*nternational Organization for Standardization*), entre outros. Estes modelos podem ser aplicados em concordância

com o modelo de desenvolvimento de aplicações da empresa, bem como com o processo/método de execução de testes. Podem ser necessários alguns ajustamentos, mas cada vez mais é importante estruturar os processos das empresas e garantir a qualidade das aplicações (Fritzsche & Keil 2007).

Por uma questão de definição do tema desta dissertação, uma vez que, no estudo empírico, o método de testes é baseado no modelo CMMI será elaborada uma pequena contextualização de seguida sobre este modelo.

2.2.2. CMMI – *Capability Maturity Model Integration*

Os CMMs são modelos que representam de forma simplificada a realidade. Estes modelos começaram a ser desenvolvidos por Walter Shewhart que aplicou os seus princípios de controlo de qualidade estatístico de forma a melhorar os processos nas empresas. Entenda-se por processo uma sequência de passos realizados para atingir um determinado objetivo. Watts Humphrey, Ron Radice e outros aperfeiçoaram os princípios e começaram a aplica-los na IBM (*International Business Machine*) no *Software Engineering Institute* (SEI) (Software Engineering Institute 2010).

O SEI tem seguido a premissa de gestão de processos "a qualidade de um sistema ou produto é altamente influenciada pela qualidade do processo usado para o desenvolver e manter" e definiu CMMs que incluam esta premissa (Software Engineering Institute 2010, p.5).

CMMs "descreve um caminho de melhoria evolutiva desde processos *ad-hoc* e imaturos até processos disciplinados e maduros com melhor qualidade e eficiência" (Software Engineering Institute 2010, p.5).

O projeto de CMMI foi despoletado devido há existência de múltiplos CMMs. O CMMI combina um conjunto de modelos numa *framework* única. Esta *framework* providencia a estrutura necessária para produzir modelos, treino e componentes de avaliação (Software Engineering Institute 2010).

Existem vários níveis de CMMI pelo qual as empresas têm de atingir para melhorar os seus processos gradualmente e existem dois caminhos, chamados representações, pelos quais as empresas podem optar conforme o seu interesse, para progredir nesses níveis (Software Engineering Institute 2010).

Através da representação contínua as empresas atingem níveis de capacidade e através da representação por fases atingem níveis de maturidade (Quadro 6).

Nível	Continuous Representation Capability Levels	Staged Representation Maturity Levels
Nível 0	Incomplete	
Nível 1	Performed	Initial
Nível 2	Managed	Managed
Nível 3	Defined	Defined
Nível 4		Quantitatively Managed
Nível 5		Optimizing

Quadro 6 – Níveis de maturidade e capacidade
Adaptado de Software Engineering Institute (2010)

Na representação contínua as empresas selecionam uma área de processo em particular (conjunto de práticas relacionadas numa determinada área de negócio, que, quando implementadas coletivamente, satisfazem objetivos considerados importantes para melhorar essa mesma área) para a melhorar, sendo que podem existir áreas de processo em diferentes níveis de capacidade (Software Engineering Institute 2010).

Na representação por fases as empresas selecionam um conjunto de áreas de processo para as melhorar, sendo que, a empresa só atinge um determinado nível de maturidade

quando todas as áreas de processo estão nesse nível (Software Engineering Institute 2010).

2.2.3. Técnicas de teste

Existem duas principais técnicas de teste mais referenciadas pelos autores: Caixa Branca e Caixa Preta.

O tipo de teste Caixa Branca é uma técnica de teste estrutural onde os Casos de Teste são desenhados com base no código fonte. Esta técnica foca-se no mecanismo interno de uma aplicação, analisando o fluxo de dados (Myers 2004; IEEE Standards Board 1990). É uma técnica usada para realizar *debugging* no código, encontrar erros tipográficos aleatórios e descobrir assunções incorretas na aplicação.

Os Casos de Teste são construídos de forma a (1) garantir que todos os caminhos independentes de um módulo foram executados pelo menos uma vez, (2) executar todas as decisões lógicas de forma positiva e negativa, isto é, com parâmetros que resultem no resultado esperado e que resultem numa mensagem de erro. Os Casos de Teste devem ainda (3) executar todos os ciclos (*loops*) nas suas fronteiras e com os limites operacionais e (4) executar estruturas de dados internos para assegurar a sua validação (Pressman 2001).

Dentro das técnicas de teste de Caixa Branca temos as estáticas (*Desk Checking, Code walkthrough* e *Formal Inspections*) e as estruturais (*Control flow/Coverage testing, Basic path testing, Loop testing, Data flow testing)* (Nidhra & Dondeti 2012).
As estáticas envolvem apenas o código fonte e não os binários e executáveis. Normalmente, os testes são efetuados antes do código estar completo e o principal

objetivo é verificar se o código está de acordo com os requisitos funcionais, com o desenho técnico e *standards* (Saglietti et al. *apud* Nidhra & Dondeti, 2012).
As estruturais, por sua vez, têm em conta o código, bem como a sua estrutura, desenho interno, e como foi codificada a aplicação (Nidhra & Dondeti 2012).

O tipo de teste Caixa Preta contrasta com o tipo de teste Caixa Branca na medida em que não tem em conta o código fonte e o mecanismo interno da aplicação, focando-se apenas no *output* gerado a partir de um determinado *input* e condições de execução (IEEE Standards Board 1990). A aplicação é vista como uma caixa preta, ou seja o utilizador não sabe o que é efetuado por detrás do interface da aplicação. Os Casos de Teste são desenhados com base na Especificação de Requisitos e os dados introduzidos têm como objetivo obter um determinado *output* que será confrontando com o resultado esperado (Myers 2004).

Esta técnica é aplicada na aplicação final, mas a sua utilização inicia-se na fase de análise dos requisitos. Os testes realizados através desta técnica não devem ser efetuados pelos próprios programadores, ao contrário das técnicas de caixa branca que podem ser executadas pelos próprios programadores (Nidhra & Dondeti 2012). A sua utilização pretende encontrar erros que se encaixem nas categorias (1) funções incorretas ou inexistentes, (2) erros de interface, (3) erros nas estruturas de dados ou na base de dados externa, (4) erros de *performance* ou comportamento e (5) erros de iniciação e encerramento.
Os Casos de Teste devem ser desenhados de forma a (1) minimizar o número total de Casos de Teste necessários para validação da aplicação e (2) que digam algo sobre a presença ou ausência de classes de erros, em vez de dizer algo apenas sobre um erro associado a um teste específico (Pressman 2001).

Em termos de processo de verificação e validação, as técnicas de Caixa Branca são usadas para verificação ("Estamos a construir o produto bem?") e nas fases iniciais do ciclo de produto. As técnicas de Caixa Preta são usadas para validação ("Estamos a construir o produto certo?") e são aplicadas depois das técnicas de Caixa Branca (Williams 2006).

Tipo de Teste	**Especificação**	**Âmbito**	**Técnica de Teste**	**Quem efetua os testes?**
Teste de Componentes ou Unitário	Desenho de baixo nível Estrutura de código atual	Pequena unidade de código	Técnica de teste de Caixa Branca	Programador que escreveu o código
Testes de Integração ou de Interface	Desenho de baixo nível Desenho de alto nível	Classes múltiplas	Técnica de teste de Caixa Branca e Preta	Programador que escreveu o código
Testes de Sistema ou Funcionais	Desenho de alto nível Análise de Requisitos	Aplicação completa em vários ambientes	Técnica de teste de Caixa Preta	*Tester* independente
Testes de Aceitação	Análise de Requisitos	Aplicação completa no ambiente do cliente	Técnica de teste de Caixa Preta	Cliente
Testes de Regressão	Documentação alterada Desenho de alto nível	Qualquer um dos anteriores	Técnica de teste de Caixa Preta e Branca	Programadores e *testers* independentes

Quadro 7 – Cruzamento entre técnicas e tipos de Testes
Adaptado de Williams (2006)

Outra técnica de testes não tão usual é a técnica baseada em erros que utiliza o histórico de erros para desenhar os Casos de Teste. Não é uma técnica tão aprofundada na teoria como as anteriores e também não é tão utilizada na prática. Dentro desta técnica temos a abordagem *Error Seeding* e *Mutation Analysis* que, de forma resumida, consistem em comparar a aplicação com "mutantes", ou seja, outras versões da aplicação, de forma a avaliar a adequação de um conjunto de Casos de Teste (Barbosa et al. n.d.).

2.2.4. Critérios de escolha do método de testes

Foram realizadas várias pesquisas em relação aos critérios de escolha de um método de testes, mas não foi encontrada muita matéria neste âmbito. As pesquisas elaboradas tiveram como palavras chave *criteria for selection of testing method(s), criteria for selection of testing strategies, criteria for selection of testing procedures, how to choose a testing method, selection of testing methods/strategies/procedures*. Outros derivados destas expressões foram introduzidos nos motores de busca: www.scholar.google.pt; http://www.b-on.pt/; http://search.proquest.com/.

A única informação obtida prende-se com os critérios de escolha de uma determinada técnica de testes. Segundo Barbosa et al. (n.d.) os principais critérios de escolha de uma técnica são o custo e a eficácia. Para além destes dois critérios, os autores referem que, através da análise da cobertura dos Casos de Teste realizados, é possível melhorar o processo de execução de testes. A cobertura dos Casos de Teste, por outras palavras, significa o número de possibilidades (Casos de Teste) que foram testadas na aplicação. Quanto mais possibilidades forem testadas mais probabilidade existe de se detetarem erros, por outro lado, pretende-se minimizar o tempo despendido no processo de execução de testes, o que implica avaliar também os Casos de Teste, de forma a verificar se não existem redundâncias (Ratzmann & Young 2003).

3. Metodologia

3.1. Método de abordagem

A metodologia usada foi o Estudo de Caso numa instituição bancária com o objetivo de estudar o seu método de Testes Funcionais. Foi enviado um e-mail a solicitar a colaboração e autorização para efetuar o Estudo de Caso na instituição, nomeadamente,

recolher informação sobre o método de execução de Testes Funcionais implementado e efetuar duas entrevistas: uma a um colaborador interno e outra ao responsável pelo departamento de testes. Foi acordado com a instituição que o seu nome permaneceria anónimo e todos os exemplos seriam de carácter abstrato.

Existem várias definições de o que é um Estudo de Caso, mas observamos que o significado do conceito é muito semelhante para os vários autores que estudam esta temática. Segundo Yin (2003), esta metodologia é usada quando uma pergunta "como" ou "porquê" está a ser questionada sobre um conjunto de eventos contemporâneos, sobre os quais o investigador tem pouco ou nenhum controlo. Pode recorrer-se ao seguinte quadro para escolher a metodologia mais adequada ao estudo a realizar:

Estratégia	**Forma da questão de investigação**	**Requer o controlo de eventos comportamentais?**	**Foca-se em eventos contemporâneos?**
Experimental	Como, porquê?	Sim	Sim
Pesquisa	Como, o que, onde, quantos, quanto?	Não	Sim
Análise de arquivo	Como, o que, onde, quantos, quanto?	Não	Sim/Não
História	Como, porquê?	Não	Não
Estudo de caso	Como, porquê?	Não	Sim

Quadro 8 – Metodologias
Adaptado de Yin (2003)

Tendo em conta que a principal questão de investigação começa com a palavra "como" e devido às razões já enumeradas na introdução (o tema ser atual e a não necessidade de observar eventos comportamentais) foi escolhida esta metodologia.

A empresa em questão foi escolhida para realizar este Estudo de Caso devido à autora desta dissertação ser uma colaboradora da empresa e devido à disponibilidade

demonstrada pela própria instituição em facultar a informação necessária e permitir elaborar o Estudo de Caso sobre o seu método de Testes Funcionais.

3.2. Técnicas de recolha de dados

As técnicas de recolha de dados consistiram na elaboração de duas entrevistas semi estruturadas que foram gravadas e transcritas. Uma das entrevistas foi realizada a um dos responsáveis pelo departamento de testes e a outra uma colaboradora interna desse mesmo departamento. Foram, também, recolhidos documentos e por fim, recorreu-se à observação direta do método de testes implementado.

A entrevista realizada ao responsável pelo departamento de testes com o objetivo de perceber quais foram os critérios de escolha do método de testes funcionais atual e também, para perceber, do seu ponto de vista, se o método escolhido é o mais adequado, quais são os seus pontos fortes e fracos e ainda perceber se é possível aplicar métodos de teste diferentes a diferentes aplicações e, por fim, saber se existia algum processo de avaliação dos resultados obtidos da implementação do método em questão.

O guião desta entrevista foi elaborado numa perspetiva de, primeiro, perceber em que modelo estava inserida a atividade de testes funcionais e depois perceber as razões pelas quais se escolheu o método de testes funcionais atual. Para complementar, tentou-se perceber os pontos fortes e fracos do método, bem como sugestões de melhoria. Uma vez que, através de observação direta percebeu-se que existia muita documentação tratada manualmente, também foi colocada uma questão de forma a perceber se a instituição tinha ponderado em utilizar uma ferramenta de suporte ao método e até que ponto acharia essa utilização vantajosa.

Por fim, foram colocadas outras 3 questões direcionadas para as restantes questões de investigação relacionadas com a utilização de métodos diferentes para diferentes aplicações e a avaliação do método atual.

A segunda entrevista foi realizada a um colaborador interno, que trabalha atualmente no departamento de testes e acompanhou a implementação do método. Esta entrevista teve o objetivo de conhecer a perspetiva de uma pessoa que trabalha com o método no dia-a-dia e que também efetuou testes funcionais antes da implementação do método. Pretendeu-se perceber se acha que é o método mais adequado, perceber quais são os pontos fortes e fracos e possíveis pontos de melhoria.

O guião desta entrevista foi elaborado de forma a perceber a importância dada aos testes funcionais pelos colaboradores da instituição e perceber os benefícios e os pontos fracos da implementação do método atual. Adicionalmente, também se pretendeu saber as vantagens do uso de uma ferramenta de suporte aos testes e procurar mais informação para as duas questões de investigação secundárias ("É possível aplicar diferentes métodos de Testes Aplicacionais a diferentes aplicações?" e "Como se devem avaliar os resultados obtidos da implementação de um determinado método?").

Foi disponibilizada pela instituição alguma documentação sobre o método de testes funcionais e também foram disponibilizados alguns documentos usados no próprio método.

A última técnica de recolha de dados usada foi a observação direta do próprio método de forma a complementar as informações obtidas através das entrevistas e documentação sobre o funcionamento do método de testes funcionais.

4. Estudo de Caso

4.1. Contextualização

O Estudo de Caso foi realizado numa empresa pertencente a um grupo no ramo da banca com origem e sede em Portugal. O grupo bancário em questão teve início no século XV na era da monarquia, tendo como principal propósito ajudar o povo Português em tempos difíceis de produção. No século XX, monárquicos e republicanos juntaram-se para construir a instituição conhecida hoje sob a forma de uma sociedade cooperativa de responsabilidade solidária ilimitada. Ao longo do século a instituição cresceu em termos geográficos por Portugal.

Atualmente a instituição conta com cerca de 700 balcões, 5000 colaboradores, mais de 400 000 associados e cerca de 1 200 000 clientes. O seu grupo é constituído por um conjunto de empresas especializadas, onde se destacam a caixa central como órgão de estrutura, supervisão e orientação, bem como a instituição de representação cooperativa e prestadora de serviços especializados ao grupo.

O ciclo de desenvolvimento de aplicações é efetuado quase na sua totalidade por uma empresa do grupo, que diz respeito aos serviços de informática, nomeadamente desenvolvimento e manutenção das aplicações do grupo. Esta empresa tem como principais objetivos a maximização da eficácia e eficiência na prestação de serviços partilhados pela instituição e, também a otimização da utilização das infra-estruturas que servem de suporte às tecnologias de informação e ao desenvolvimento de sistemas de informação do grupo.

4.2. Descrição do método

O departamento de Testes Funcionais é constituído por uma equipa interna que tem como principal competência a supervisão e acompanhamento de uma outra equipa externa, contratada através de *outsourcing.* O método de Testes Funcionais atual foi implementado entre o ano de 2006 e 2007, tendo como base o CMMI (representação por fases). Desde então, o processo tem sido alvo de algumas melhorias, apesar da certificação não estar validada atualmente.

O principal impulso da empresa na adoção de um método de testes foi a necessidade de padronização dos vários métodos e procedimentos de trabalho que existiam. A empresa optou por este método devido à tecnologia disponível (pois o método tem como principal suporte ferramentas do Microsoft Office), às competências possuídas pelos seus recursos, devido à decisão de se certificar em CMMI e também às tendências no mundo empresarial na altura.

Por norma, o método de testes é aplicado da mesma forma independentemente do modelo de desenvolvimento (que no caso é principalmente o modelo *Waterfall* – em cascata) e da tipologia do âmbito em que a aplicação é testada, se no âmbito de um projeto ou pedido. Sendo que, uma aplicação é testada sob a forma de projeto quando se trata de uma aplicação nova, construída de raiz, enquanto um pedido é uma alteração a uma aplicação já existente e portanto o objetivo é apenas testar uma componente da aplicação.

Este método é constituído por 3 fases (Estimativa, Planeamento e Execução de testes), sendo que a empresa contratada, através de *outsourcing,* pode participar em todas elas ou em parte delas em diferentes projetos. Dentro de cada fase existe um conjunto de tarefas que devem ser realizadas sempre que é necessário a execução de Testes

Funcionais. Conforme estejamos perante um projeto ou pedido podem existir tarefas que não sejam necessárias realizar. Na figura 6 estão descritas as tarefas inerentes às várias fases.

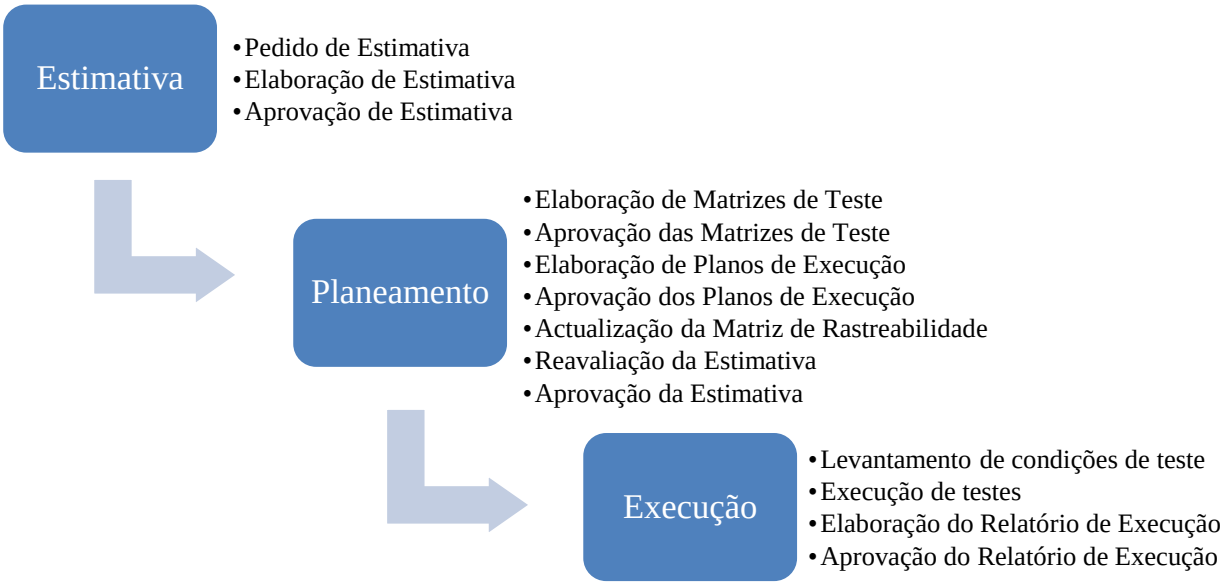

Figura 6 – Fases e respetivas tarefas do método de execução de Testes Funcionais

Na primeira fase é elaborada uma estimativa, que se traduz no cálculo do esforço (em termos de tempo e recursos) necessário para a realização das fases de planeamento e/ou execução de testes. A estimativa pode ser efetuada tanto pela equipa de testes, em *outsourcing*, ou por um dos colaboradores internos. O cálculo do esforço tem como base os recursos disponíveis/necessários para realizar as tarefas, bem como a documentação sobre o projeto em causa (Especificação de Requisitos e Análises Funcionais) e eventuais imprevistos que possam surgir.

Depois da aprovação da estimativa elaborada, pelo responsável interno do departamento, é iniciada a fase de Planeamento. Esta fase é constituída por 3 tarefas: Elaboração de Matrizes de Teste, Elaboração de Planos de Execução e Atualização da Matriz de Rastreabilidade.

A fase de Planeamento pressupõe que toda a documentação esteja na versão "Aprovada" de forma a permitir a sua análise. Depois de analisada toda a documentação é elaborada a Matriz de Teste que facilita o mapeamento dos Casos de Teste necessários para garantir o cumprimento de cada requisito presente na documentação entregue (Especificação de Requisitos e/ou Análises Funcionais).

Nesta fase é definida a estrutura dos Planos de Execução. Por norma, cada plano representa uma funcionalidade ou uma componente do programa e a sua constituição divide-se em Produtos, Cenários de Teste e, por fim, Casos de Teste. Um Produto pode ser definido como uma divisão da componente que se está a tratar no Plano, por exemplo, a componente "Registo de Cliente" pode ter como Produtos "Registo de Clientes – Particulares" e "Registo de Clientes - Empresas". Cada Produto, por sua vez, subdivide-se em Cenários e estes, finalmente subdividem-se em Casos de Teste. Para o Produto "Registo de Clientes-Particulares" podemos ter como cenários "Caixa com privilégio XPTO" e "Outras caixas". Em relação aos Casos de Teste, estes validam um ou mais requisitos, através de um fluxo (Básico ou Alternativo), que consiste num conjunto de passos a seguir. Dentro de cada Requisito há um conjunto de pontos a testar (a aplicação mostra o campo X, com as opções Y, Z e W?)

Após a aprovação das Matrizes de Teste, mais uma vez pelo responsável interno do departamento ou pelo colaborador interno que está a acompanhar o processo, inicia-se a tarefa de Elaboração de Planos de Execução.

Um Plano de Execução é um documento baseado nas Matrizes de Teste que especificam todos os passos necessários para testar um determinado requisito ou requisitos, de forma a atingir um determinado objetivo (Validar a componente X, ou a funcionalidade Y…), em cada Caso de Teste.

Uma vez realizados os Planos de Execução é elaborada a Matriz de Rastreabilidade, cujo *layout* é disponibilizado pela mesma área que elaborou a Especificação de Requisitos. A Matriz de Rastreabilidade tem como principal objetivo localizar todos os requisitos da Especificação de Requisitos nos Planos de Execução realizados. Assim, para cada requisito é indicado em que plano(s) se encontra e consequentemente em que Produto, Cenário de Teste e Caso de Teste.

A fase de Planeamento é finalizada com a reavaliação da estimativa realizada inicialmente e é disponibilizado a aplicação para testes. Posteriormente, é iniciada a fase de Execução de Testes com a tarefa de levantamento de condições de teste.

O levantamento de condições pressupõe a existência de pré condições nos Planos de Execução realizados. Estas condições passam por gerar informação com determinadas características de forma a verificar diferentes comportamentos da aplicação, como por exemplo, disponibilização de vários perfis de utilizador com privilégios diferentes. É importante fazer este levantamento antes da execução de testes propriamente dita de forma a poupar tempo e perceber as várias variáveis que impactam no comportamento da aplicação.

Assim que forem reunidas todas as condições de teste procede-se à execução de testes, seguindo os Planos de Execução. Idealmente, quem elaborou os Planos de Execução não os executa na fase de Execução de Testes de forma a manter o processo o mais objetivo e imparcial possível. A mesma regra se aplica para quem desenvolve aplicação, pois caso contrário existiriam conflitos de interesse. Uma vez que o processo de testes é encarado como um processo "destrutivo" seria contra producente alocar recursos de desenvolvimento de aplicações, que constroem, a testar.

Antes da realização de qualquer teste é criado um documento denominado PdS – Ponto de Situação, que para além de permitir a consulta do progresso de execução de testes, permite o controlo e rastreio das incidências (erros) detetadas ao longo do processo. O PdS é um documento de controlo que também é enviado ao cliente periodicamente de forma a dar visibilidade do progresso da execução de testes.

À medida que os testes são executados é necessário recolher "provas" de que a aplicação está a ter o comportamento esperado e conforme o que está especificado. Essas "provas" são registadas em forma de *printscreen* dos vários écrans da aplicação e são inseridas num ficheiro denominado Ficheiro de Evidências.

Quando ocorre um comportamento diferente do esperado, ou seja, quando não se verifica um determinado output conforme o Plano de Execução é necessário registar uma incidência (erro) para que a situação seja corrigida, na respetiva ferramenta criada para o efeito. Se o erro encontrado não for bloqueante, ou seja, se permitir prosseguir com o fluxo do Caso de Teste, este é finalizado de forma a verificar se não existe mais nenhuma inconsistência.

Finalmente a última tarefa da última fase é a elaboração do Relatório de Execução que consiste no resumo de todas as ações relativas ao processo de Verificação e Validação.

4.3. Ferramentas de suporte

Não existe uma ferramenta que suporte o método do início ao fim. Contudo, ao longo do tempo, foram desenvolvidas pequenas ferramentas para auxiliar o trabalho das equipas de testes.

Uma das ferramentas desenvolvida pretende ajudar na componente técnica e estética dos Planos de Execução. O Pds também tem um ficheiro de suporte em Excel com

gráficos que demonstram a evolução do trabalho realizado. Existe ainda, uma ferramenta com algumas funcionalidades relacionadas com a verificação de *standards*. Finalmente existe uma aplicação para registar as incidências, onde é possível comunicar entre as diferentes áreas envolvidas no projeto/pedido.

4.4. Processo de avaliação

Não existe um processo de avaliação propriamente dito para avaliar o método de testes em si. É realizada uma avaliação no âmbito apenas dos projetos e existe uma avaliação de performance anual dos colaboradores internos. Tanto um como o outro são efetuados com base em métricas definidas.

4.5. Pontos de melhoria do método

Embora exista um método de Testes Funcionais existem alguns pontos de melhoria identificados ao longo do Estudo de Caso. Esses problemas provocam perda de eficiência e eficácia no processo de execução de Testes Funcionais, pois existem redundâncias, perdas de tempo e produtividade, o que leva também à desmotivação das pessoas.

Alguns desses pontos de melhoria prendem-se com:

- Falta de comunicação entre os vários departamentos (Departamento de negócio, Departamentos de desenvolvimento e Departamento de Validação e Verificação);
- Demasiado foco na documentação o que origina excesso de burocracia;
- Falta de uma ferramenta de suporte ao método completo.

Existem várias fases do ciclo de vida de um produto, o que implica a intervenção de vários departamentos. No método de testes adotado é necessária a comunicação entre estes, de forma a esclarecer dúvidas e resolver determinadas situações. Devido ao

elevado número de pessoas envolvidas no processo e devido às diversas especialidades que cada um é responsável, por vezes, é difícil que todos os recursos envolvidos consigam estar em sintonia. A comunicação deveria ser um meio de resolução eficaz das dificuldades enfrentadas, contudo existe algum trabalho a realizar nesse sentido.

Um dos objetivos da implementação do método atual foi a pdronização quer das tarefas realizadas quer da documentação utilizada e produzida. A empresa tem trabalho para atingir este objetivo, mas por vezes, o seu alcance torna o método contraproducente, devido às constantes alterações que são necessárias realizar para manter toda a informação atualizada. Como ao longo de cada ciclo de produto existem várias alterações à ER, bem como pedidos de alteração pela parte do cliente, os planos também são constantemente alterados. Além de poder ser necessário reanalisar e refazer as estimativas, estas tarefas consomem tempo e os frutos obtidos nem sempre são compensatórios.

Por fim, um outro ponto de melhoria indicado pelos colaboradores internos é a implementação de uma ferramenta de suporte à execução de testes. Devido ao aumento de complexidade dos projetos e devido ao tempo despendido em cada tarefa, impõem-se a necessidade de agilização do método de execução de testes. Foram apontadas várias vantagens que uma ferramenta traria, tais como monitorização constante da evolução do trabalho realizado; facilidade em documentar e proceder a alterações a essa documentação sempre que necessário; automatização de testes que também seria benéfico para a realização de testes de regressão; entre outros benefícios. Esta ideia ainda não foi concretizada devido ao elevado investimento que implica e também à necessidade de adaptação dos recursos humanos e do método de trabalho à ferramenta.

5. Conclusão e Investigações Futuras

Os Testes Aplicacionais têm-se revelado cada vez mais importantes para as empresas e a sua evolução tem seguido no sentido de, cada vez mais, ser uma tarefa independente e imparcial em relação à tarefa de programação de uma aplicação.

Há medida que o conceito de Testes Aplicacionais se foi construindo surgiram técnicas e métodos de forma a tornar esta tarefa mais eficiente, uma vez que padronizar processos torna mais fácil a identificação de falhas nas aplicações.

Não foram encontrados métodos específicos para Testes Funcionais o que constituiu uma limitação nesta dissertação. Ainda assim, constatou-se que em termos práticos, no Estudo de Caso, o método usado é constituído por um conjunto de boas práticas, técnicas, ferramentas e processos aplicados e melhorados ao longo do tempo, tendo como base o CMMI.

Foi elaborado um quadro-resumo (Anexo 4) onde foi feito um cruzamento entre a teoria e a prática de forma a identificar até que ponto, na prática se recorre a conceitos teóricos. Constatou-se que a perspetiva relativa aos Testes Funcionais presente na instituição vai de encontro à perspetiva de Myers (2004) que defende que os Testes Aplicacionais (neste caso, Testes Funcionais) têm o objetivo de encontrar erros e no limite melhorar a qualidade da aplicação.

Em relação às técnicas, ferramentas e práticas é aplicada, principalmente a técnica Caixa Preta e, embora se reconheçam vantagens na utilização de uma ferramenta de suporte aos testes, não existe uma ferramenta que suporte o processo/método de execução de testes completo, mas sim ferramentas que ajudam em algumas fases do processo. O método aplicado é baseado no CMMI (representação por fases) e segue o processo descrito por Sommerville (2011) na Figura 5.

Respondendo à questão colocada inicialmente ("Como devem as empresas escolher o método de Testes Aplicacionais mais adequado ao seu negócio?") com base no Estudo de Caso realizado a escolha de um método de Testes Aplicacionais pode depender de variáveis como a cultura da empresa (competências dos recursos, valores…), as boas práticas, a tecnologia adotada, os custos e os próprios modelos de desenvolvimento de *software.*

No Estudo de Caso verificou-se que o método adotado sofria ligeiras alterações consoante se tratasse de um projeto ou pedido, mas o método mantém-se o mesmo. Conclui-se que é difícil aplicar diferentes métodos a diferentes aplicações, uma vez que a adoção de um método tem como principal objetivo padronizar os vários processos ligados à tarefa de testar.

Para estudos futuros seria interessante aprofundar os temas relacionados com as duas outras questões colocadas inicialmente: "É possível aplicar diferentes métodos de Testes Aplicacionais a diferentes aplicações?" e "Como se devem avaliar os resultados obtidos da implementação de um determinado método?". O Estudo de Caso elaborado serve também como ponto de partida para um estudo futuro onde se comparem métodos de diferentes empresas de forma a perceber se um mesmo método pode ser aplicado a empresas com aplicações distintas.

Referências

Aalders, R., 2002. IT Outsourcing: Making it work.

Alexandre, V. & Cunha, F., 2008. Outsourcing de Tecnologias de Informação. , pp.1–15.

Barbosa, E.F. et al., Introdução ao Teste de Software £. , pp.1–49.

Beck, K., 1999. *Extreme Programming Explained* 1ª ed.

Bernardo, P.C. et al., 2008. A Importância dos Testes Automatizados. , 1(3), pp.54–57.

ESA Board for Software Standardisation and Control, 1995. Guide to software verification and validation. , (1).

Foundation Certificate in Software Testing, 2010. ISEB / ISTQB. , 3.

Fritzsche, M. & Keil, P., 2007. Agile Methods and CMMI - Compatibility or Conflict? *Information Software Engineering Journal*, 1(1).

Gelperin, D. & Hetzel, B., 1988. The Growth Of Software Testing.

IEEE Computer Society, 2005. *IEEE Standard for Software Verification and Validation*, New York.

IEEE Standards Board, 1990. IEEE Standard Glossary of Software Engineering Terminology.

Li, E.Y., 1990. Software Testing In A System Development Process: A Life Cycle Perspective. *Journal of Systems Management*, 41(August), pp.23–31.

Myers, G.J., 2004. *The Art of Software Testing* 2ª ed. I. Word Association, ed., New Jersey.

Nidhra, S. & Dondeti, J., 2012. Black Box and White Box Testing Techniques - A Literature Review. , 2(2), pp.29–50.

Porto Editora, 2013. *Dicionário de Língua Portuguesa da Porto Editora*.

Pressman, roger S., 2001. *Software Engineering - a practitioner's aprroach* 5ª ed. I. McGraw-Hill, ed., New York.

R. Jeevarathinam, A.S.T., 2010. Towards Test Cases generation from software specifications. , 2(11), pp.6578–6584.

Ratzmann, M. & Young, C. de, 2003. *Software Testing and Internationalization* Galileo Press GmbH, ed.

Scott Ambler, 2002. *Agile Modeling: Effective Pratives for eXtreme Programming and the Unified Process*.

Software Engineering Institute, 2010. *CMMI® for Acquisition, Version 1.3 CMMI-ACQ, V1.3*.

Sommerville, I., 2011. *Software Engineering* 9ª ed. Pearson, ed.

Williams, L., 2006. Testing Overview and Black-Box Testing Techniques.

Yin, R. K., 2003. *Case Study Research: Design Methods.* 3 ed. London: Sage Publications.

Anexos

Anexo 1 – Guiões de entrevistas

Guião da entrevista ao responsável do departamento de testes aplicacionais

1. Qual é o método/modelo de desenvolvimento de aplicações implementado na empresa?
2. O método de execução de testes aplicacionais está implementado desde quando e é baseado em quê? Quais foram os critérios de escolha desse método?
3. Considera benéfico o uso de um método para a execução de testes? Quais são as suas vantagens?
4. Quais são os pontos de melhoria desse método?
5. Considera benéfico que a atividade de planeamento não seja efetuada pelos programadores?
6. Em que medida este método tem-se tornado mais eficiente?
7. Acha que o uso de uma aplicação de suporte ao método seria benéfico? E a automatização de testes?
8. Seria possível aplicar uma metodologia de testes diferente dependentemente do tipo de projeto ou aplicação?
9. É feita uma avaliação dos resultados obtidos do uso desse método? Se sim, como é feita?
10. Alguma vez foi implementada uma melhoria no método através dos resultados da avaliação?

Guião da entrevista a um dos colaboradores internos que tem contacto direto no dia-a-dia com o método de execução de testes aplicacionais

1. Considera importante a execução de testes funcionais? Porquê?
2. Desde quando tem contacto com este método?
3. Trabalhou com o método anterior? Como era o seu trabalho com esse método?
4. Qual acha mais benéfico para o seu trabalho?
5. Existe algum aspeto do método usado anteriormente que considerasse importante para uma melhor execução do seu trabalho?
6. Quais são, para si, os principais benefícios e pontos de melhoria do método usado atualmente?
7. Acha que o uso de uma aplicação de suporte ao método seria benéfico?
8. Sabe que existe uma avaliação dos resultados obtidos através da implementação do método? Acha que essa avaliação é eficiente?
9. Alguma vez foi implementada uma melhoria no método usado atualmente?

Anexo 2 – Transcrição da entrevista ao responsável do departamento de testes aplicacionais

1. **Qual é o método/modelo de desenvolvimento de aplicações implementado na empresa y?**

Na componente do sistema considerado central de suporte ao negócio bancário, obedece mais a uma lógica do modelo em cascata, *Waterfall*. Relativamente à abordagem mais ágil, esta está mais relacionada com desenvolvimentos em outros ambientes, designadamente em ambientes distribuídos e *Web Based*. Portanto aí procura-se utilizar essas metodologias que, em bom rigor, em projetos recentes tem-se vindo a procurar adotar e tem-se vindo a provar que não é assim tão ágil como isso. Para ser ágil pressupõe que haja rigor nos levantamentos iniciais, haja rigor na documentação dessas mesmas funções, seja ao nível de alteração, seja ao nível de novas necessidades e de que em matéria de desenvolvimento se tome isso em linha de conta. Portanto pressupõe entregas parciais, mas com a garantia que decorrente dos testes não surjam situações de erro e de ocorrências.

2. **O método de execução de testes aplicacionais está implementado desde quando e é baseado em quê? Quais foram os critérios de escolha desse método?**

Utilizamos este método há 6/7 anos, desde 2006/2007. Incide nos nossos testes, na componente não só de testes funcionais, mas também na perspetiva de utilizador final. O nosso processo de verificação e validação está muito intrincado ao CMMI e obedeceu a uma lógica de certificação até determinada altura. Existe uma lógica de validação dessa mesma certificação que deixou de se suceder e portanto os níveis de exigência são maiores, mas eu diria que foi uma coisa associada à outra. No processo de avaliação que foi feito na altura, das diversas metodologias de que deviam ser adotadas pela empresa, não só na parte de testes, mas também ao nível de processos de aquisições, ao nível do processo de desenvolvimento e, designadamente, da gestão de configurações, ao nível da gestão de projetos, direi que foram dados passos nesse sentido. Foi entendido, na altura, que a melhor abordagem face à realidade da área financeira seria utilizar o modelo CMMI e o processo de verificação está intrincado a esse modelo. Neste momento, a área de testes não tem a sua certificação validada, mas na altura pretendia-se atingir uma certificação de nível 2. Apesar do surgimento de novas tecnologias e métodos creio que este método continua a ser atual. É evidente que também estes métodos um dia perdem validade porque também depende da forma como a empresa se quer estruturar. Se a empresa se quer estruturar de uma outra forma, se fizer determinadas opções tecnológicas provavelmente, poderá também isso conduzir à adoção de outros métodos. Antigamente o método utilizado era que a área de desenvolvimento fazia tudo, este é um movimento recente e nem todas as empresas têm uma área à parte.

3. **Considera benéfico o uso de um método para a execução de testes? Quais são as suas vantagens?**

O processo em si não se encerra apenas nos testes funcionais e devia ser mais vasto. Realisticamente, há um conjunto de ações que deverão ocorrer ao longo do ciclo de vida de desenvolvimento e a essas ações devem existir um trabalho de verificação e validação. Em cinco anos, todo o processo cingia-se apenas a uma especificação de requisitos, mas que era uma dupla perspetiva, era um misto entre especificação de requisitos e já uma aproximação a especificações funcionais. Durante essa fase, há todo um processo de verificação da forma como se estão a especificar os requisitos do cliente, em alguns casos em termos funcionais, de forma a verificar em que medida é que é o pretendido pelo cliente e isso é assegurado. Há determinadas ações, provavelmente no momento já de desenho, em que não estava verdadeiramente formalizado o processo. Não havia esse compromisso da empresa, portanto era uma lacuna que a empresa sentia e parte do processo não era assegurado. Nos dias de hoje, essa situação está a ser feita mas cinge-se apenas a projetos. Não há um carácter de obrigatoriedade para projetos de

menor dimensão, da mesma forma de que ações de manutenção evolutiva, que surja após a implementação de determinadas aplicações e, que na altura tiveram documentos de análise funcional de suporte, não são atualizados. O processo de análise funcional é recente, mas em contrapartida o processo de verificação já tem alguns anos. Está quase nos primórdios de quando esta empresa foi reconfigurada há seis, sete anos nesta parte, e nós desde então temos procurado paulatinamente, à medida que vamos participando em projetos, procurar uma base que nos permita em situações futuras, sobre essas mesmas aplicações dar outro grau de qualidade. Tendo como base os requisitos, nos casos em que exista também uma análise funcional associada, procuramos especificar um conjunto de casos de teste que nos permita fazer uma abordagem adequada e eficaz e que procure eliminar um conjunto de situações que não vão de encontro ao que é pretendido pelo cliente, garantindo que o produto final vai de acordo às necessidades do cliente.

4. **Quais são os pontos de melhoria desse método?**

Em matéria das metodologias é importante que haja uma comunhão de objetivos, ou seja, tem-se de ver que, para cada uma das fases que compõe os projetos, todas elas têm a sua importância. Todas elas têm de dar o seu contributo, portanto não estamos a falar aqui em fases autónomas, mas sim interligadas. Daí a conveniência de estarmos todos conscientes de que a informação que se está a passar deve ser partilhada e permite, não só assegurar a boa execução das nossas atividades, mas também ser útil para atividades subsequentes, por isso é importante haver essa comunhão de interesses, objetivos e partilha de informação que nos permita cada vez mais sermos mais eficientes e garantirmos maior qualidade no serviço que prestamos.

5. **Considera benéfico que a atividade de planeamento não seja efetuada pelos programadores?**

Aqui na casa ainda não esta suficientemente claro quais são as fronteiras da área de desenvolvimento e as fronteiras do departamento que assegura os testes funcionais. Quando vim para cá sempre preconizei a ideia de que no âmbito do desenvolvimento tem de ser acionado alguns testes e há dois que são fulcrais, que são os testes unitários e os testes integrados, porque em situações em que haja interfaces, interligações com diversas aplicações e que envolva, consequentemente, distintas áreas de desenvolvimento tem que haver testes que assegurem isso. Nós estamos sempre no plano dos testes funcionais, é certo que cada vez mais procuramos ir um bocadinho mais além, mas temos uma dificuldade acrescida. Não podemos ir um pouco mais além, porque não sendo elaborada documentação, não nos dão essa possibilidade. Contudo pode ser-nos dada, em última instância, a responsabilidade: "Porque é que isto não foi atendido?". A área dos testes está num plano de independência, incluse existe o *independent verification and validation*, que é uma área independente. Porque o argumento de desenvolvimento nesta empresa é de que não lhes damos muita informação para não os influenciar nos vossos testes. Eu acho que é importante ter uma área autónoma, que não tenha qualquer vinculo com nenhuma das outras áreas, de forma a que não esteja a influenciar. Que siga o seu caminho e que faça a sua abordagem e é isso que nós temos procurado fazer, mas não vivemos sozinhos, dai a conveniência de atuarmos em rede, partilhando informação, mas temos a nossa autonomia, a nossa independência.

6. **Em que medida este método tem-se tornado mais eficiente?**

Inicialmente, havia o conceito de que se tinha de fazer um plano para um projeto. Eu tinha um plano único, portanto era um plano que chegava a atingir centenas quase milhares de páginas, para fins de manutenção era complicado. Da mesma forma que em matéria de desenvolvimento, também na parte de especificação de casos de testes, nós temos que nos basear em determinados princípios basilares que é a mutualidade, facilidade de manutenção, estarmos sempre preocupados com isso. Quando nós especificamos casos de teste temos que ver, segregar ou agregar as coisas, criar alguma modularidade, preocupação numa manutenção futura, portanto

haver aqui vantagens ou sinergias em fazermos as coisas desta forma para, em fases futuras tirarmos dai dividendos.

7. **Acha que o uso de uma aplicação de suporte ao método seria benéfico? E a automatização de testes?**

Sim, desde que vim para cá, sempre fui apologista de ferramentas, só que as ferramentas para além dos custos relacionados com a aquisição, quando se pretende comprar ou adquirir tem-se que pensar não só no investimento nessa ferramenta, mas na sua utilidade. É necessário potenciar a sua utilização para justificar o investimento que é feito. Para isso, nós temos de ter a base, comprar uma ferramenta sem termos as bases asseguradas é contraproducente, é um mau investimento. Quando vim para ca, havia um processo em curso, antes da minha vinda, em 2006, que era a aquisição de uma ferramenta de gestão de requisitos, para depois fazer a ponte e ser usada nas fases subsequentes, nomeadamente na abordagem aos testes funcionais. Contudo esse processo nunca evoluiu. Quando eu cheguei estava parado e continua parado nos dias de hoje. Não obstante isso havia aqui uma intenção e, nos dias de hoje, acho ainda muito atual. Só que as ferramentas têm o tal problema, requererem determinados *skills*. Tem que haver esses *skills* que consigam manter essa ferramenta e que têm de ter, não só conhecimentos funcionais, de negócio, mas também algumas aptidões técnicas, não os tendo é complicado. No entanto não basta ter apenas conhecimento, porque o conhecimento, não pode residir apenas na cabeça das pessoas, o conhecimento tem de estar refletido em documentação e portanto se isso não for feito rapidamente conduz ao mau investimento porque as situações não se podem garantir, não se consegue sistematizar e automatizar, designadamente a questão dos testes. Acredito que se venham dar passos nesse sentido, mas requer tempo. Por outro lado, a adoção dessa ferramenta não é imperativo que seja uma ferramenta única que cubra todo o espectro do ciclo de desenvolvimento de *software*, o que é importante é que haja interligação entre as mesmas e portanto mais uma vez o tal objetivo comum que é aqui aspetos principais que é a documentação das aplicações, o manter permanentemente atualizadas, ter um repositório único e que se possa a partir dai tirar partido dessa informação. Acho que podíamos retirar vantagens de uma ferramenta, como gravar um conjunto de *Test Cases* e reproduzir em qualquer momento, se houver necessidade corrente de novos pedidos de manutenção, atualizar o que houver a atualizar e puder reproduzir. Na abordagem dos testes não há uma pretensão de testar tudo à infinésima possibilidade, portanto há aqui um momento a partir do qual os testes que venhamos a fazer já não temos grandes ganhos de os fazer. Temos é de fazer uma abordagem criteriosa, abordagem essa que nos dê uma garantia de que temos já um nível de confiança e um nível de qualidade já apreciável. Com a adoção da ferramenta permitir-nos-ia saber em que medida estamos a testar todas as situações. A adoção de ferramentas não será útil apenas para se servir de repositório para os requisitos ou para uma automatização dos testes. Com um conjunto de testes efetuados permitia saber, face às componentes de *software* que estão a ser submetidas a testes, qual foi o grau de cobertura. As ferramentas têm a vantagem da sistematização também e permite termos uma informação diária ao minuto de qual é o estado de arte em termos do nosso processo, em termos de execução de testes, portanto dar visibilidade de um conjunto de incidências identificados ao longo do processo; permite saber qual é o grau de realização até ao momento e ter toda a rastreabilidade. Obviamente com ferramentas o processo é mais fácil.

8. **Seria possível aplicar uma metodologia de testes diferente dependentemente do tipo de projeto ou aplicação?**

Nos pedidos o que procurámos fazer foi atualizar o plano no repositório referente ao projeto onde incide o pedido e transformar o mesmo plano, mas numa realidade de pedido. A abordagem aí em nada difere é só a questão da dimensão. Quando se trata de um pedido avulso e na ausência do tal dito repositório, tem-se ido por outro caminho. Baseia-se no mesmo princípio mas é mais simples. Sempre cumprindo o mesmo objetivo que é, identificando todas as situações, embora sejam *Test Cases* menos detalhados, á luz dos requisitos que foram enunciados, garantir que o produto que foi desenvolvido dá resposta a esses mesmos requisitos, por isso o princípio de validação esta sempre subjacente. Nós aí independentemente do suporte

que temos de base, da origem das necessidades e da forma como ela é implementada a nossa abordagem acaba por ser a mesma. Só não é quando não adotamos em nada o nosso processo, ou seja é dizer que não especificamos casos de testes e baseamo-nos apenas na informação que nos é disponibilizada, seguimos esses requisitos e conforme o método antigamente testa-se sem planos formais. E daí tem os seus consequentes prejuízos porque poderá advir problemas. Mesmo tendo evidências, elas poderão não ser suficientemente elucidativas de forma a que qualquer elemento que amanhã venha a ter necessidade de testar consiga perceber a abordagem seguida. É importante especificar as coisas, planear, documentar, manter vivo o documentar e depois pô-lo em prática que é executá-lo.

9. **É feita uma avaliação dos resultados obtidos do uso desse método? Se sim, como é feita?**

Sim, existem métricas associadas ao projeto e métricas de organização indexadas a objetivos, quer ao nível de empresa quer ao nível de áreas. Métricas são indicadores para medir diversos itens que se dividem em indicadores de projeto, organização e gestão de desempenho. Da mesma forma que existem métricas face à qualidade dos requisitos que estão a ser elaborados. Só do ponto de vista do desenvolvimento uma das métricas é o número de incidências detetadas e que deram azo à correção de software em função do número de pontos de observação que foram especificados, para isso pressupõe a especificação dos casos de teste. É evidente que se detalharmos algo o rácio é muito baixo e portanto falsamente podemos dizer que o *software* é muito bom, é sempre uma grande discussão. O tal projeto pioneiro no nosso processo que é a substituição da plataforma de balcões, o sentimento e a perceção das equipas de projeto era de que o desenvolvimento não era dos melhores em termos da qualidade, no entendimento do fornecedor, que era quem desenvolvia o *software*, dizia que sim, porque eram largas centenas de incidências que foram objeto de correção mas estávamos a falar para um universo de pontos de observação de largas dezenas centenas de milhar e isso era uma ínfima parte. Logo a conclusão era que o projeto era de boa qualidade. Tirando esse aspeto, apesar de dados factuais mas com alguma subjetividade, essas métricas são recolhidas e daí chegasse à conclusão que a área de desenvolvimento "ok cumpriu os objetivos" ou não porque também depende das metas que são fixadas. Há um aparte que é a dependência do rigor no registo das coisas. Se não houver rigor tamos já, à partida, a enviesar os resultados. Do nosso lado é mais sob o ponto de vista de um conjunto de manutenções corretivas que possam surgir decorrente do processo de implementação e de disponibilização em ambiente de produção. Também muitas vezes é subjetivo porque há um conjunto vasto de alterações e há várias disponibilizações em ambiente de produção para a mesma funcionalidade decorrente de distintos pedidos. À parte disso não é feito verdadeiramente uma avaliação, são dados mais estatísticos que nalguns deles dá-se o caso de estarem indexados a objetivos de equipa. No âmbito do nosso processo temos uma métrica que é 80% dos projetos, ao longo de um ano, têm de estar com um nível de 20% de rácio entre erros face ao ponto de observação. Por norma, os projetos não são muitos e para os casos em que nós intervimos estamos mais ou menos na média, são poucos os que passam e provavelmente é a situação em que devemos redefinir as metas sob pena que estamos a dar uma ideia errada.

10. **Alguma vez foi implementada uma melhoria no método através dos resultados da avaliação?**

A maioria das situações que são identificadas prende-se com a articulação entre as áreas que intervêm no processo de desenvolvimento de *software*, para isso tem que haver alguém acima destas áreas que de facto, entenda por bem que tem que haver melhorias nesse sentido. Casos em que haja uma dependência, que envolvam mais do que uma área, torna-se complicado ou tem sido difícil implementar essas melhorias. A título de exemplo, nós temos um objetivo desde o ano passado que é a identificação de ações de melhoria, com uma particularidade, podemos dar sugestões de melhoria para outros processos onde não participamos, mas não conta para os nossos objetivos. Não obstante essa situação no ano passado foi dada uma sugestão de melhoria que não foi atendida curiosamente com o argumento de que enquanto não houver ferramenta de

requisitos acham que não deve ser feita. Não víamos como fosse impossível fazê-lo, se já feito para projetos. Mais uma vez o apelo a que tivéssemos a informação o mais completa possível e interligada, isto a partir do momento em que houve o surgimento de um processo de análise funcional. Qualquer sugestão que apresentemos no relatório de execução que envolva alterações de outras áreas há uma forte probabilidade e assim tem decorrido, de não serem atendidas. Temo-nos cingindo a procurar melhorar cada vez mais o nosso processo, no princípio de melhoria, procurando simplifica-lo, enriquece-lo, ir um pouco mais além. Algo que foi feito este ano, em prol dos testes de aceitação, foi criar um suporte eletrónico para o registo de incidências.

Anexo 3 – Transcrição da entrevista a um dos colaboradores internos que tem contacto direto no dia-a-dia com o método de execução de testes aplicacionais

1. **Considera importante a execução de testes funcionais? Porquê?**

Acho que os testes funcionais são o mais importante em termos de todos os testes. Sou um bocadinho suspeita, porque a minha área são os testes funcionais, na ótica do utilizador final e o que nós queremos sempre garantir é que a aplicação corresponda sempre áquilo que o cliente está a pedir. No entanto, acho que sem se efetuar testes funcionais, por muitos testes de carga, de regressão, de integração que se façam, se não chegarmos a este nível de pormenor não íamos conseguir garantir os requisitos.

2. **Desde quando tem contacto com este método?**

De uma forma mais assertiva foi por volta de 2007, contudo era pretensão da empresa, a partir do momento em que nós somos uma empresa certificada em CMMI, de nível II na área dos requisitos, certificar também a área de testes numa fase, que salvo erro, de nível 5, que é o ultimo estádio. Só que para já isso ainda não aconteceu. Temos vindo a trabalhar nesse sentido, ou seja, a preparar a eventualidade de chegarmos a essa fase e este trabalho que temos vindo a fazer é precisamente que se de hoje para amanha se quisermos entrar na certificação, estarmos minimamente preparados. Foi com base nisso que se fez formação com um empresa que elaborou os atuais *templates* e isso esta a cargo da área de metodologias e standards com que trabalhamos diretamente, nós e todas as outras áreas, sendo que só a área de requisitos esta certificada. No entanto, deveríamos já ter documentação funcional e técnica também standardizada e não está. Só a meio do ano passado se começou a trabalhar sobre isso. Nós começámos a par e passo com os requisitos, ou seja, entramos muito antes que qualquer área. Tivemos muitas dificuldades porque sem especificação de requisitos, sem documentação funcional e técnica bem trabalhada não se conseguem garantir grandes *Test Cas*es e cenários de teste da forma como gostaríamos.

3. **Trabalhou com o método anterior? Como era o seu trabalho com esse método?**

Sim, desde 2000 que estou na área de testes. Não havia um método standardizado, cada um trabalhava ao seu método. A área existia, eram feitos testes da mesma forma era garantida a qualidade da mesma forma, se calhar não nos perdíamos tanto no pormenor do técnico, mas sim mais com o pormenor funcional. É certo que se calhar se fossemos procurar um repositório ou documentação, não iriamos encontrar, porque não estava standardizada, mas cada um de nós conforme os pedidos iam surgindo, tínhamos as nossas evidências, nem que fossem em papel e tínhamos ficheiros de Excel com que trabalhávamos. Antigamente, e nós trabalhávamos também mais para a vertente de pedidos, pois a casa não estava tao vocacionada para projetos, o que também facilitava em termos de execução de testes tínhamos *templates* de Excel e muito pouco mais. Tínhamos relatórios, os relatórios finais eram feitos e pontos de situação também se faziam. Tudo isso se fazia, não estavam era standardizados.

4. **Qual acha mais benéfico para o seu trabalho?**

É difícil dizer…o meu trabalho passou a ser diferente quando se mudou de método. Enquanto na altura eu testava, fazia o ciclo de testes na sua totalidade e como a dimensão passou a ser maior e o tempo de resposta diminuiu eu passei, no fundo a supervisionar uma equipa de testes. Como a dimensão aumentou nós não conseguimos perceber o que está a ocorrer ao mesmo tempo. Hoje estou a executar um projeto e algo foi definido de uma determinada forma e depois surge um pedido que vem alterar isso. É certo que já tínhamos os planos do projeto, mas às vezes o projeto é testado por uma pessoa e os pedidos por outra e é difícil perceber o funcionamento da aplicação de uma forma panorâmica. A especificação de requisitos que devia estar sempre atualizada não ajuda porque não temos um repositório e rastreabilidade de requisitos. Com a ferramenta de gestão já seria possível por exemplo dizer que um determinado requisito é comum a n projetos.

5. **Existe algum aspeto do método usado anteriormente que considerasse importante para uma melhor execução do seu trabalho?**

Há uma grande diferença em relação ao método anterior. O departamento de testes decidia a entrada em produção. Nós é que dizíamos que não estava em condições para entrar em produção e neste método não é isso que acontece. Não somos nós que decidimos, nós dizemos que os testes estão concluídos conforme o especificado. Muitas vezes, nós somos os primeiros a dizer que, na nossa opinião, não entraria, mas não temos esse poder. Alguém decide que é para entrar, da forma como entender, não sendo o departamento de testes a decidir isso. Enquanto que, no método antigo nós garantíamos até ao utilizador final as condições e às vezes fazíamos testes conjuntamente com as áreas de cliente. Agora não. Nós garantimos que os testes funcionais estão de acordo com o que está especificado, mas depois há uma aceitação formal da parte do cliente. No entanto não quer dizer que este método seja pior, pois se o cliente pediu algo esse pedido tem que estar conforme aquilo que o cliente pediu. No entanto, como em qualquer outra empresa, o cliente pediu uma coisa, mas depois quer acrescentar outra e o processo é mais moroso.

6. **Quais são, para si, os principais benefícios e pontos de melhoria do método usado atualmente?**

Na minha opinião, mantendo o método que temos hoje, começa a tornar-se complicado trabalhar sem uma ferramenta de testes. Nós não temos ferramentas de gestão de testes que é uma coisa que faz muita falta para quem trabalha com este tipo de metodologia. Pois é preciso gerir, quer os testes quer os requisitos, porque só assim se consegue criar rastreabilidade entre eles e reaproveitá-los.

O *template* de testes era uma das coisas que deveria mudar, porque é um *template* muito complexo e que demora muito tempo a elaborar. Contudo também acho que há aplicações que registam tudo o que estamos a fazer a nível da aplicação, seria sem duvido muito mais rápido. Em relação a este método a única coisa que eu retirava era o plano de verificação e validação porque com tanto planeamento estamos constantemente a alterar o plano. Até poderia existir mas não nos moldes em que existe, porque ele é feito aquando da caracterização do projeto e é obrigatório que o façamos. Quando se está a fazer uma caracterização de projeto não se tem o planeamento na sua totalidade e depois corremos o risco de estar sempre a alterá-lo. Aquilo que mais me faz consumir tempo é os planos, a verificação que se faz, a atualização que se tem de fazer, o versionamento que se faz, o ter que se validar o que se faz. Isso torna o processo moroso, eu sei que se tivéssemos uma ferramenta de gestão de testes as coisas não seriam assim. Para se ter a ferramenta de testes também temos de ter um trabalho feito, e a manutenção também não é fácil. Com certeza haverá pros e contra mas uma ferramenta viria a ajudar. Em termos de método propriamente dito acho que há formalismo a mais no que diz respeito ao versionamento mas também sei que é necessário ter o repositório. A aplicação que temos para

gerir os documentos não é o sitio mais correto para ser fazer isso, devia haver uma aplicação de gestão documental e devia haver menos formalismos.

Temos uma coisa que não tínhamos antes que é a documentação standardizada.

7. **Acha que o uso de uma aplicação de suporte ao método seria benéfico?**

Esta questão foi respondida anteriormente. A ferramenta suporte e gestão de testes foi indicada como um ponto de melhoria.

8. **Sabe que existe uma avaliação dos resultados obtidos através da implementação do método? Acha que essa avaliação é eficiente?**

Sim, sei da sua existência. Acho que não é muito eficiente, devido à pouca objectividade e fácil manipulação dos resultados das métricas que são usadas para efetuar a avaliação. Até porque quando a aplicação chega à fase de testes funcionais já não deveria ter muitos erros e como não são efectuados, na maioria das vezes, outros testes, nós somos "obrigados" a ter em conta erros não só funcionais, mas também técnicos.

9. **Alguma vez foi implementada uma melhoria no método usado atualmente?**

Sim, objectivamente conseguimos implementar as melhorias a que nos propusemos no ano passado, a nível dos testes de aceitação.

Anexo 4 – Cruzamento entre a teoria e prática

Entrevista 1 – Entrevista ao responsável do departamento de testes aplicacionais

Entrevista 2 – Entrevista a um dos colaboradores internos que tem contacto direto no dia-a-dia com o método de execução de testes aplicacionais

Área	Afirmações	Autor	Questões
Conceito e objetivos dos Testes Aplicacionais	- Atualmente, existe um processo de Verificação e Validação que pretende aumentar a qualidade das aplicações ao longo do seu ciclo de vida. - Testar é o processo de executar um programa com o principal objetivo de encontrar erros. Para além disso, testar tem também como propósito acrescentar valor à aplicação, aumentando a sua qualidade.	IEEE Computer Society (2005) e Myers (2004)	3 - Considera benéfico o uso de um método para a execução de testes? Quais são as suas vantagens? (Entrevista 1) 6 - Em que medida este método tem-se tornado mais eficiente? (Entrevista 1) 1 - Considera importante a execução de testes funcionais? Porquê? (Entrevista 2)
Processo/Métodos de Testes Aplicacionais	- Neste modelo existe uma sequência de fases e a passagem de uma para a outra	Pressman (2001) R. Jeevarathinam (2010), Bernardo et al. (2008),	1 - Qual é o método/modelo de desenvolvimento de aplicações

	acontece quando a primeira está finalizada - A geração automática permite aumentar a rapidez do desenho de Casos de Teste, permite elaborar Casos de Teste mais complexos, permite efetuar testes de regressão mais amplos, ou seja verificar todos os Casos de Teste depois de efetuada uma alteração e permite, também, aumentar a cobertura de testes efetuados à aplicação. - Estas ferramentas são usadas com o principal objetivo de melhorar a eficiência do processo de execução de testes, através da automatização de tarefas. - Estes modelos podem ser aplicados em concordância com o modelo de desenvolvimento de aplicações da empresa, bem como com o processo/método de execução de testes.	Foundation Certificate in Software Testing (2010) e Fritzsche & Keil (2007)	implementado na empresa? (Entrevista 1) 2 - O método de execução de testes aplicacionais está implementado desde quando e é baseado em quê? Quais foram os critérios de escolha desse método? (Entrevista 1) 7 - Acha que o uso de uma aplicação de suporte ao método seria benéfico? E a automatização de testes? (Entrevista 1) 6 - Quais são, para si, os principais benefícios e pontos de melhoria do método usado atualmente? (Entrevista 2) 7 - Acha que o uso de uma aplicação de suporte ao método seria benéfico? (Entrevista 2)
Técnicas de testes	- Os testes realizados através desta técnica (caixa peta) não devem ser efetuados pelos próprios programadores, ao contrário das técnicas de caixa branca que podem ser executadas pelos próprios programadores	Nidhra & Dondeti (2012)	5 - Considera benéfico que a atividade de planeamento não seja efetuada pelos programadores? (Entrevista 1)