TESTES FUNCIONAIS DE APLICAÇÕES WEB E MOBILE

Adilson Gomes Monteiro Pereira

Relatório de Estágio | Mestrado em Informática

Especialização em Sistemas de Informação

Orientação: Prof. Doutor João Lima Pinto

Maio, 2017

DEGI | UPT UNIVERSIDADE PORTUCALENSE

Adilson Gomes Monteiro Pereira

Testes Funcionais de Aplicações Web e Mobile

Relatório submetido à Universidade Portucalense Infante D. Henrique, para cumprimento dos requisitos necessários à obtenção do grau Mestre em Informática – Especialização em Sistemas de Informação, elaborado sob a orientação do Prof. Doutor João Lima Pinto.

Departamento Economia, Gestão e Informática

Maio, 2017

"A força de vontade deve ser mais forte do que a habilidade"

Muhammad Ali

Agradecimentos

Em primeiro lugar gostaria de agradecer à empresa Itsector e a toda a sua administração que me permitiram a realização do estágio proporcionando um excelente ambiente de trabalho e condições para aprender, desenvolver e melhorar as minhas competências técnicas e humanas.

Agradeço em particular ao meu orientador Dr. João Lima Pinto que sempre me prestou apoio mostrando total disponibilidade. Também queria agradecer em especial a minha coorientadora Elisabete Pires (responsável pela equipa de testes) pelo papel importante que teve nos 6 meses de estágio, pela forma como fez a gestão do meu estágio, pelos conhecimentos técnicos e princípios que me foram passados. Além disso, agradeço ainda toda a confiança que depositaram em mim, inserindo-me em projetos importantes, desde muito cedo. Agradeço também, toda a disponibilidade para ajudar e apoiar em todas as ações dentro da empresa.

Agradeço à tester Joana Mota pelo apoio e conhecimento que me passou contribuindo na minha evolução em cada projeto; também quero prestar o meu agradecimento aos programadores pela consideração e respeito que sempre demostraram por mim em projetos em que estivemos envolvidos e aos meus colegas de trabalho pela forma como me ajudaram neste período.

Agradeço aos meus pais, que certamente são os principais responsáveis pela realização deste mestrado, possibilitando condições financeiras e o acreditar sempre nas minhas capacidades. Queria agradecer à minha namorada, aos meus irmãos, meus amigos e todos meus familiares que contribuíram para conclusão desta etapa.

Por fim resta agradecer a Universidade Portucalense, a todos seus docentes, colegas e colaboradores por contribuírem para a realização dos meus estudos, e em especial à Dr.ª Paula Morais que teve um papel importante no meu percurso académico, por toda a motivação e incentivo que me foi passado, por todos os conhecimentos que me transmitiu e sobretudo por todos os conselhos dados em momentos difíceis. Por fim, deixo também um agradecimento especial à Dr.ª Filomena Lopes e ao coordenador dos mestrados Dr. Fernando Morreira por todo o apoio prestado durante este tempo.

Resumo

O presente relatório descreve o estágio realizado como parte integrante do mestrado em Informática - especialização em Sistemas de Informação da Universidade Portucalense, na empresa Itsector - Sistemas de Informação, SA.

Nos últimos anos, a indústria de software, tem empregado cada vez mais recursos na busca pela qualidade de produtos e na redução dos custos de desenvolvimento e manutenção. A atividade de teste é essencial no desenvolvimento de um software, pois apesar das técnicas, métodos e ferramentas empregados, produto ainda podem ocorrer erros.

A empresa Itsector segue a abordagem de desenvolvimento ágil oferecendo produtos inovadores aos clientes e por isso as atividades de testes precisam de estar incluídas no processo de desenvolvimento desde o início, acompanhando todo o ciclo de vida de um software.

Os testes de software são classificados em vários tipos e são aplicados em diferentes níveis. Um dos tipos de testes aplicados na empresa são os testes funcionais, mais concretamente, a utilização da técnica de testes manuais às aplicações web e mobile. As atividades de testes são planeadas e coordenadas pela equipa de testes e são registadas na ferramenta de colaboração e gestão de projetos Team Foundation Server (TFS). A ferramenta TFS é utilizada e partilhada por todos os intervenientes no processo de desenvolvimento das aplicações.

O objetivo do estagio consistia na realização de tarefas de testes às aplicações desenvolvidas e realização da análise relativamente aos processos de testes aplicados na empresa com a finalidade de melhorar os processos de testes desta área.

O estagiário foi inserido na equipa de testes interagindo diretamente com a equipa de desenvolvimento. Estas interações permitiram ao estagiário adquirir experiencia e conhecimento sobre os processos de testes aplicados na empresa, pelo que este no fim do estágio, elaborou uma análise critica aos processos de testes apresentando soluções para melhoria desses processos.

Palavras-chaves: Testes funcionais, Testes de software, Testes de aplicações web, Testes de aplicações mobile, Team Foundation Server, Testes em desenvolvimento ágil.

Abstract

This report describes the stage, performed as an integral and final part of the Master in Computer Science - specialization in Information Systems from the University Portucalense in Itsector company - Information Systems, SA.

In recent years, the software industry has increasingly employed resources in the search for product quality and in the reduction of development and maintenance costs. The testing activity is essential in the development of software that despite the techniques, methods and tools employed, errors in the product may still occur.

The company Itsector follows the agile development approach offering innovative products to customers and therefore the testing activities need to be added to the development process from the beginning, accompanying the entire software life cycle. Software tests are classified by various types and are applied at different levels. One of the types of tests applied in the company are the functional tests, but concretely the use of the technique of manual tests to the mobile and web applications. The testing activities are planned and coordinated by the test team and are recorded in the Team Foundation Server (TFS) collaboration and project management tool. The TFS tool is used and shared by all stakeholders in the application development process.

The aim of the internship was to carry out tasks of testing the applications developed and performing the analysis regarding the testing processes applied in the company with the purpose of improving the testing processes in this area.

The trainee was entered into the test team interacting directly with the development team. These interactions will allow the trainee to acquire some experience and knowledge about the processes of tests applied in the company, reason why this at the end of the stage will elaborate a critical analysis to the processes of presenting solutions for the improvement of these processes.

Keywords: Functional tests, Software testing, Web application testing, Mobile application testing, Team Foundation Server, Agile development tests.

ÍNDICE

Índice de Ilustrações

Índice de Figura

Definições e Acrónimos

Debuging - processo de encontrar e reduzir defeitos num aplicativo de software ou mesmo em hardware.

Extreme Programming - metodologia ágil de desenvolvimento que se destina a melhorar a qualidade do software e a capacidade de resposta às mudanças nas necessidades dos clientes.

Open source - modelo de desenvolvimento que promove um licenciamento livre para o design ou modificação do produto.

Product Backlog – lista de todas as funcionalidades desejadas para um produto.

Proxy - servidor que age como um intermediário para requisições de clientes, solicitando recursos de outros servidores.

Rapid Application Development - modelo de processo de desenvolvimento de software iterativo e incremental que enfatiza um ciclo de desenvolvimento extremamente curto.

Release - uma nova versão atualizada de um determinado software.

Responsive - técnica de programar um site de forma a que os elementos que o compõem se adaptem automaticamente, a qualquer dispositivo.

Scrum - framework de desenvolvimento ágil para planeamento e gestão de projetos de software.

Tester - individuo que faz testes às aplicações.

TFS - (Team Foundation Server) - plataforma de colaboração para projetos de desenvolvimento de software que permite integração entre todos os participantes do projeto, permitindo a gestão de projetos, qualidade, gestão do código fonte.

User story - definem os aplicativos, requisitos e elementos que as equipas de programação precisam criar.

1. Introdução

1.1 Contextualização

A crescente utilização de sistemas baseados em computador, em praticamente todas as áreas de atividade humana, levou a que a engenharia de software evoluísse significativamente nas últimas décadas procurando estabelecer técnicas, critérios, métodos e ferramentas para a produção de software, procurando qualidade, tanto do ponto de vista do processo de produção como do ponto de vista dos produtos gerados (Sommerville, 2015).

No processo de desenvolvimento de software, a maioria dos defeitos é causada pelos seres humanos e, apesar do uso dos melhores métodos de desenvolvimento e ferramentas, os erros permanecem frequentemente nos produtos, o que torna a atividade de teste fundamental durante o desenvolvimento de um software (Neto, 2007).

Atividades agregadas sob o nome de garantia de qualidade de software têm sido introduzidas ao longo de todo o processo de desenvolvimento, entre elas atividades de verificação, validação e teste, com o objetivo de minimizar a ocorrência de erros e riscos associados (Barbosa, et al., 2010). Os testes de software fazem parte do processo de garantia da qualidade do software, medindo a qualidade do produto em termos de defeitos nos requisitos funcionais e não-funcionais (Pinheiro, 2015).

Garantir a qualidade do software representa um conjunto de atividades de prevenção que devem ser aplicadas e geridas ao longo de todo o processo de desenvolvimento, envolvendo revisões técnicas formais, múltiplas fases de teste, controlo da documentação de software e das mudanças nos procedimentos, a fim de garantir ao cliente que o software desenvolvido estará de acordo com as especificações e atendendo às suas necessidades (Pressman, 1995). O controlo da qualidade de software é um desafio enorme devido à complexidade dos produtos e às diversas dificuldades relacionadas com o processo de desenvolvimento, que podem envolver questões humanas, técnicas, burocráticas, de negócio e políticas (Carvalho, 2010).

Existem diferentes abordagens para o desenvolvimento de software e uma delas tem ganhado bastante destaque, a abordagem ágil. Seja qual for a metodologia de desenvolvimento implementada, as atividades de testes de software são de enorme importância.

Os testes de software no modelo tradicional de desenvolvimento utilizam requisitos para derivar casos de testes e, a partir destes, implementa-se a execução de testes manuais e automatizados, enquanto que, no desenvolvimento ágil trabalha-se com pequenas iterações com entregas ao final de cada iteração, e por isso os testes devem ser executados de forma continua e integrada desde o início do projeto, fazendo parte integrante de todo o processo de desenvolvimento (Bortoluci & Duduchi, 2015).

O tamanho do projeto e a quantidade de pessoas envolvidas no processo de desenvolvimento são dois possíveis fatores que podem aumentar a complexidade de testes de software e, consequentemente aumentam a probabilidade de defeitos. Assim, a ocorrência de falhas é inevitável (Devmedia, 2010). Testar software tornou-se numa tarefa mais complexa devido ao grande número de novas linguagens de programação, sistemas operativos, plataformas, hardware e tecnologias que continuam a evoluir ao longo do tempo (Myers, 2004). O principal objetivo dos testes de software é revelar a presença de erros no produto. Isto é, o teste bem-sucedido é aquele que consegue determinar casos de teste para os quais o programa em teste falha (Barbosa, et al., 2010).

O processo de teste geralmente envolve uma mistura de testes manuais e automatizados. No teste manual, um tester executa o programa com alguns dados de teste e compara os resultados com as suas expectativas anotando e reportando as discrepâncias aos desenvolvedores do programa, enquanto os testes automatizados são codificados num programa que é executado cada vez que o sistema em desenvolvimento é testado.

Existem diversas ferramentas de apoio à atividade de testes, as quais ajudam a automatizar as atividades do processo de teste. As ferramentas quando utilizadas corretamente trazem melhorias para a equipa de testes e para toda a equipa de um projeto de software, permitindo cooperação entre toda a equipa, a atualização de todos envolvidos no projeto, redução no tempo entre a identificação e a correção de erros e maior segurança quando se efetuam alterações no código (da Silva & Moreno, 2011).

1.2 A empresa

Fundada em 2005, a IT Sector (http://www.itsector.pt/pt/) – Sistemas de Informação, S.A., é uma empresa portuguesa de desenvolvimento de software, criada com o objetivo de oferecer ao mercado Soluções de Sistemas de Informação de elevado valor acrescentado, contando com 300 colaboradores.

O Grupo IT Sector integra atualmente: a ebankIT, uma empresa especializada em produtos multicanal direcionada para o mercado financeiro; a Bitmaker, uma empresa de desenvolvimento de software e de serviços de outsourcing especializada na entrega de projetos críticos; e com o objetivo de fortalecer as alianças com a África, a IT sector formou parceria estratégica com Angola CPCÁfrica e é a acionista da CPCÁfrica Moçambique.

Ilustração 1 - Grupo Itsector

A capacidade de implementar projetos completos e complexos e os serviços de Outsourcing e de desenvolvimento em regime Nearshore, são as principais valências da ITSector, sendo que a partir de localizações estratégicas em Portugal, desenvolve software para mais de 20 países como: Angola, Moçambique, Reino Unido, Luxemburgo, Polónia, Rússia, França, Islândia, Dinamarca, Macau, Timor, África do Sul, entre outros com inúmeras soluções em mercados financeiros, telecomunicações e saúde.

1.2.1 Cultura e Valores

A cultura da ITsector é o resultado de um conjunto de princípios e valores que inspiram as suas políticas e atuações. Sendo os pilares culturais os seguintes:

- Satisfação do Cliente, Promovendo a Confiança e a Credibilidade
- Ética e Responsabilidade social
- Iniciativa, Dinamismo e Inovação
- Flexibilidade e autonomia
- Desenvolvimento dos seus Recursos Humanos

E porque o êxito da ITSECTOR depende tanto da sua capacidade tecnológica como dos seus colaboradores, a ITSECTOR estimula ativamente o seu desenvolvimento e soft skills como:

- Profissionalismo
- Espírito de Equipa
- Iniciativa

1.2.2 Mercados

A empresa Itsector atua em diferentes mercados com forte presença no mercado africano com um número maior de clientes em Angola e Moçambique conforme ilustra a figura 2.

Ilustração 2 - Mercados da ITsector

1.2.3 Organigrama

Apresenta-se na figura 3 o organigrama da empresa, assinalando a área onde se enquadra o estágio. O estagiário pertence ao departamento de produção mais concretamente á área da qualidade (Quality & Assurance) e está alocado na equipa de testes.

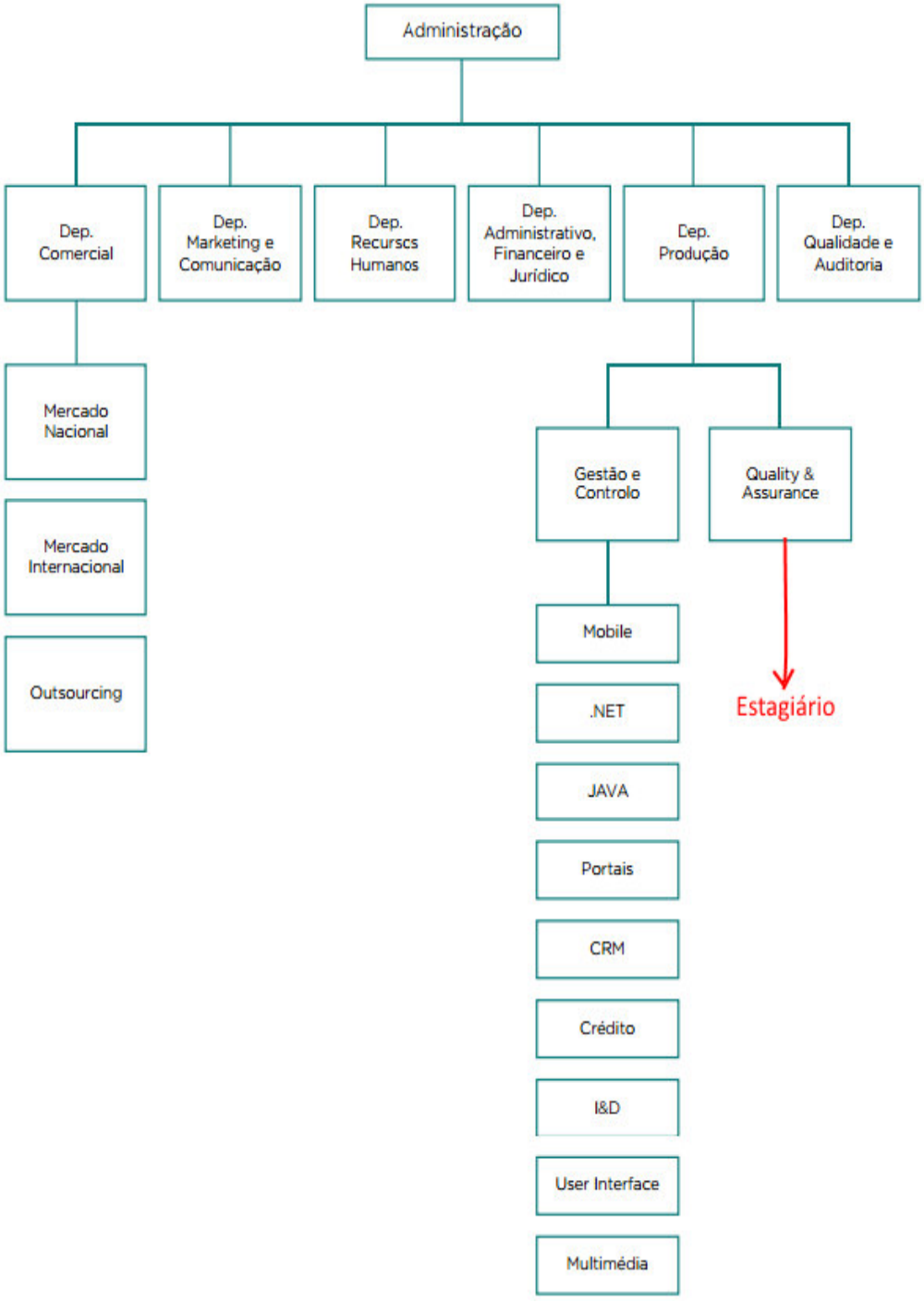

Ilustração 3 - Organigrama da ITsector

1.3 Âmbito e Objetivos

A Itsector empresa em que se insere o estágio, é uma empresa especializada em desenvolvimento de software, nomeadamente desenvolvimento de aplicações web e mobile. Sendo uma empresa que aposta forte na inovação através do uso das tecnologias de informação está em constante desenvolvimento de soluções complexas e criticas, pelo que a fase de testes é de vital importância.

Os projetos são testados cuidadosamente antes de serem entregues aos clientes, passando por aprovações por parte da equipa de teste.

No âmbito do estágio, o estagiário foi inserido em projetos de testes de aplicações web e mobile, tornando-se num novo elemento da equipa de testes com o objetivo de contribuir para a análise de processos de testes, interagindo com a equipa de desenvolvimento, procurando diminuir, significativamente, o número de erros e garantir a qualidade das aplicações.

Numa fase inicial foi necessário ao estagiário conhecer como a equipa de testes da Itsector trabalha, quando alocada a um projeto. O objetivo do estagiário foi obter o conhecimento necessários para a posição de tester, para ser capaz de elaborar casos de testes, reportar bugs e fazer a revalidação dos bugs resolvidos.

O estágio foi enquadrado na área de testes de software mais concretamente no âmbito dos testes funcionais de aplicações web e mobile.

Pretendia-se também uma contribuição do tester através da análise de processos e métodos de teste aplicados na Itsector, de forma a sugerir melhorias no processo de testes.

O projeto de estágio, possibilitará, em suma:

- Uma experiencia profissional ao nível empresarial;
- Aquisição de novas competências técnicas;
- Contacto com novas ferramentas (TFS)
- Melhorar as competências qualidades de comunicação
- Melhorar a competência de trabalho em equipa
- Integração num projeto real
- Aumentar a autonomia na resolução de problemas

- Contribuir para a análise e melhoria dos processos e métodos de teste e mesmo impactar com as equipas de desenvolvimento por forma a minimizar os erros e a facilitar os testes

Apresenta-se em anexo a calendarização das tarefas realizadas durante o estágio.

1.4 Metodologia de Desenvolvimento da Itsector

Para o desenvolvimento de software a Itsector, segue a metodologia ágil implementando princípios de *Agile* Manifesto (Itsector, 2016).

Para gestão de projetos utiliza a estratégia de desenvolvimento Scrum, incentivando todos os intervenientes a trabalhar em equipa e como uma unidade para atingir um objetivo comum (Itsector, 2016).

Nesta estratégia o desenvolvimento ocorre em partes pequenas, chamadas *sprints*, em que cada *sprint* é uma "peça" que encaixa nas que foram criadas anteriormente. Cada equipa tem associados papéis, tarefas, eventos, artefactos e regras, produzindo uma "peça" de cada vez, incentivando a criatividade e permitindo que as equipas respondam ao *feedback* e mudança.

1.5 Filosofia de Teste na Itsector

Sendo que a Itsector adota a metodologia de desenvolvimento ágil, as atividades de testes começam no inicio do projeto e continuam até à conclusão final. A equipa de testes trabalha de perto com os programadores e com a equipa de negócio para garantir que são atingidos os níveis de qualidade pretendidos.

Este processo permite que haja uma comunicação frequente entre a equipa de testes e a equipa de programação e, consequentemente, que seja dado *feedback* atempado sobre a qualidade do produto, de modo que a equipa de programação se possa focar nas funcionalidades mais importantes.

As atividades de testes são planeadas com antecedência, definindo *user stories*, e o trabalho necessário por cada uma delas durante a *sprint*. Os testes realizados na Itsector normalmente são testes funcionais e testes de segurança. Neste

documento serão abordados apenas os testes funcionais visto que é o tipo de teste que se enquadra no âmbito do estagio.

Na prática os testes funcionais são realizados manualmente, ou seja, cada tarefa de teste é uma simulação da utilização da aplicação como utilizador final, através de vários cenários e garantindo que funciona corretamente.

1.6 Estrutura do documento

O presente relatório está estruturado em 4 capítulos: O primeiro capítulo apresenta a instituição onde se realiza estágio e o seu âmbito. O segundo capítulo faz um enquadramento teórico do tema testes de software abordando alguns conceitos essenciais, como os diferentes tipos e níveis que podem ser aplicados, realçando a importância dos testes no desenvolvimento de software. Também neste capítulo é abordado o tipo de testes funcionais. O terceiro capítulo descreve todo o processo de estagio, com exemplos das principais tarefas realizadas no estagio e a forma como ocorrem as atividades de testes desde inicio dos casos de testes até aos registos dos bugs. O quarto capítulo apresenta na conclusão do trabalho, abordando algumas dificuldades encontradas e sugestão para melhoria no processo de testes aplicados na empresa.

2. Testes de Software

2.1 O que são Testes de Software

Considerado por muitos como o pai dos testes de software Glenford Myers define teste de software "como um processo, ou um grupo de processos, definidos para garantir que um código faz o que foi desenhado para fazer, e não faz nada que não foi especificado para fazer" (Farias, 2014). Na prática, e resumidamente, um teste de software verifica se o software está a fazer aquilo que foi solicitado, se não faz o que não está definido para fazer, e como se comporta em determinadas situações (esperadas e inesperadas) com objetivo de garantir a qualidade do produto.

Segundo (Sommervile, 2011), outro autor reconhecido na área de engenharia de software, os testes de software não podem demonstrar se o software é livre de defeitos ou se ele se comportará conforme especificado em qualquer situação, isto é, os testes podem mostrar apenas a presença de erros, e não a sua ausência.

2.1.1 Conceitos básicos

Relativamente à atividade de testes de software há 3 conceitos básicos importantes que serão definidos seguidamente, seguindo a terminologia padrão para Engenharia de Software do IEEE – Institute of Electrical and Electronics (IEEE, 1990):

- **Defeito** é um ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Por exemplo, uma instrução ou comando incorreto.
- **Erro** é uma manifestação concreta de um defeito num produto de software. Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa constitui um erro.

- **Falha** é o comportamento operacional do software diferente do esperado pelo utilizador. Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.

Deste modo, pode-se dizer que os **defeitos** fazem parte da aplicação propriamente dita e são causados por pessoas por exemplo, mal-uso de uma tecnologia (universo físico). Os **defeitos** podem produzir manifestação de **erros** num produto, ou seja, a construção de um software de forma diferente ao que foi especificado (universo da Informação). Por fim, os **erros** geram **falhas**, que são comportamentos inesperados num software que afetam diretamente o utilizador final da aplicação e que pode inviabilizar a utilização (universo do utilizador) de um software (Neto, 2007). A figura 4 mostra a diferença entre os conceitos.

Ilustração 4 - Conceito de testes software (Neto, 2007)

2.2 Objetivo dos Testes

Todos os tester gostariam de testar todas as funcionalidades possíveis de um programa, mas, na maioria dos casos é impossível. Um simples programa pode ter centenas ou milhares de possíveis combinações de entrada e saída. Criar casos de teste para todas essas possibilidades é impraticável ou testar uma aplicação completamente levaria muito tempo e exigiria muitos recursos humanos para ser economicamente viável (Myers, 2004).

Segundo (Myers, 2004) apesar do teste de software ser uma atividade técnica também envolve algumas considerações importantes de psicologia humana. Myres afirma que uma das principais causas de falhas de testes é o facto de que a maioria dos programadores e testers começam com uma definição errada dos seguintes termos:

1. Testar é processo de demonstrar que os erros não estão presentes;
2. O objetivo do teste é mostrar que um programa executa suas funções;
3. Testar é o processo de estabelecer a confiança de que um programa faz o que é suposto fazer.

Myres argumenta que os seres humanos tendem a ser altamente orientados para o objetivo, e estabelecer o objetivo adequado tem um efeito psicológico importante. Se o objetivo do tester ou programador for demonstrar que o programa não tem erros, então subconscientemente seremos dirigidos para este objetivo; ou seja, tendem a selecionar dados de teste que têm uma baixa probabilidade de fazer com que o programa falhe.

Por outro lado, se o objetivo é demonstrar que um programa tem erros, nossos dados de teste terão uma maior probabilidade de encontrar erros.

Para perceber melhor este termo Myres analisa-se o uso e sentido das palavras “sucesso” e “sem sucesso” num contexto particular a utilização pelos gestores. A maioria dos gestores de projeto referem-se a um caso de teste que não encontra erros, como um caso de teste executado com “sucesso”. Por sua vez, um caso de teste que falha, devido à ocorrência de erros no software, referem-se a um caso de teste sem sucesso. Um exemplo pratico de uso destas palavras é a sua utilização pelos médicos; por exemplo um paciente consulta um médico por sentir um mal-estar. O medico executa alguns exames de laboratório, se não consegue localizar o problema, não se pode chamar teste com "sucesso"; neste caso o teste é “sem sucesso”, paciente gastou dinheiro, ainda está doente, e pode questionar capacidade do medico. No entanto, se o exame de laboratório determina que o paciente tem uma doença, o teste é considerado com “sucesso”, porque o médico pode agora pode começar o tratamento adequado.

Na área de testes também pode ser aplicado esse conceito, localizar o erro e resolve-lo. É importante manter a visão de que o processo de teste serve para encontrar erros e não para provar que eles não existem.

O objetivo da atividade de testes é adicionar algum valor ao software, ou seja, aumentar a qualidade ou confiabilidade do programa, encontrando e removendo erros; portanto, não devemos testar um programa para mostrar que funciona corretamente. Em vez disso, deve-se começar com a suposição de que o programa

contém erros e, em seguida, testar o programa para encontrar o maior número possível de erros.

2.3 Importância dos Testes

Testar software é necessário porque todos nós cometemos erros, alguns desses erros não são importantes, mas outros podem custar caro ou tornar-se perigosos, por isso é preciso verificar que se produz porque erros podem acontecer a qualquer momento (ISTBQ, s.d.).

No entanto, os principais fatores da realização de testes de software são a diminuição do risco, a diminuição dos custos associados à produção e manutenção e garantir que todas as especificações e expectativas definidas e esperadas pelo cliente ou serviço são cumpridas, para que não exista a insatisfação do cliente ou serviço (Carvalho, 2010).

O teste é importante, uma vez que podem ser encontrados defeitos antes do software ser entregue ao cliente, permitindo que os defeitos possam ser identificados e corrigidos em fases iniciais de desenvolvimento.

A ausência de teste pode resultar em perdas de dinheiro, perdas humanas ou ferimentos graves além da perda de reputação empresarial. Existem inúmeros exemplos de casos reias de perdas de vidas e dinheiro por ausência de teste de software. Apresenta-se nas secções seguintes alguns exemplos.

Os testes de softwares são essências para garantir a confiabilidade do cliente em relação ao produto ou aplicação, diminuem os custos de manutenção e ajudam a garantir um melhor desempenho (ISTBQ, s.d.).

A função dos testes passa por medir a qualidade do software em termos de defeitos, tanto relativamente a requisitos e características funcionais como não funcionais, com vista a reduzir a possibilidade de ocorrência de problemas.

Atualmente a maioria das empresas de desenvolvimento de software percebe claramente o impacto do custo quando não se prioriza a importância dos testes no ciclo de vida de desenvolvimento dos produtos.

O custo da correção de um defeito tende a ser cada vez maior quanto mais tarde ele for descoberto (Myers, 2004).

2.3.1 Exemplos de casos de insucesso devido à insuficiência de testes

Descrevem-se, seguidamente, alguns casos de insucesso devido à ausência ou insuficiência de testes:

- **Quase 3ª Guerra Mundial (1983)** - O sistema de alerta precoce soviético indicou que os Estados Unidos tinham lançado cinco mísseis balísticos. Causa: um bug no software soviético falhou ao detetar reflexos solares como falsos mísseis (Ribeiro, 2012).

- **Voo Iran Air 655 1988** - No final da guerra Irão-Iraque o navio USS Vincennes disparou um míssil que derrubou um avião de passageiros iraniano. Causa: durante o confronto com iranianos no Golfo Pérsico, devido a um erro de interface, o navio atingiu acidentalmente o avião depois de o ter confundido com um avião de combate, matando 290 passageiros (Hillabin, 2012).

- **Chamadas (1990),** cerca de 60.000 clientes de "longa distância" da AT & T tentaram fazer chamadas de longa distância como de costume e não conseguiram nada. A AT & T supôs que estava sendo "hackeada" e, durante nove horas, a empresa e a polícia tentaram descobrir o que estava acontecendo. No final, a AT & T descobriu uma falha no novo software. AT & T perdeu cerca de US $ 60 milhões em tarifas de longa distância de chamadas que não foram atendidas (Maderna, 2016) .

- **Ariane 5 (1996)** - custo de $ 500 milhões, o foguete da agencia espacial europeia foi destruído segundos após seu lançamento em seu voo inaugural. Causa: O desligamento ocorreu quando o computador de orientação tentou converter a velocidade do foguete de 64-bits para um formato de 16 bits. O número era muito grande, o que resultou num erro. Quando o sistema de orientação desligou, o controlo passou para uma unidade idêntica redundante, que também falhou porque nele estava correndo o mesmo algoritmo (Ribeiro, 2012).

- **Passaportes britânicos (1999)** - A agência de passaportes do Reino Unido implementou um sistema que falhou ao emitir documentos para meio milhão de cidadãos britânicos. A agência teve que pagar milhões ao governo por danos. Causa: A agência lançou o novo sistema sem testá-lo de forma adequada ou treinar os seus funcionários. Ao mesmo tempo, uma mudança na lei exigia que todos os menores de 16 anos viajando para o exterior deveriam obter um passaporte, resultando em um aumento enorme na procura de passaportes, o que sobrecarregou o sistema (Ribeiro, 2012).

- **Bug do Milênio (1999)** - Empresas gastaram biliões com programadores para corrigir uma falha no software. Embora nenhuma falha significativa ocorresse, a preparação para o Bug do Milênio teve um custo significativo e impacto no tempo em todas as organizações que usavam sistemas informáticos. Causa: para economizar espaço de armazenamento de computador, os sistemas armazenavam anos para datas com números de dois dígitos, como 99 para 1999. Esses sistemas também interpretavam 00 para significar 1900, em vez de 2000, por isso, quando o ano de 2000 se aproximou, foi necessário corrigir este problema para evitar falhas (Ribeiro, 2012).

- **Tratamento de Cancro Mortal (2000)** - 8 pessoas mortas e 20 seriamente feridas. O software de radiação da empresa Multiparte calculou mal a dosagem de radiação que deveria ser enviada, expondo pacientes a níveis fatais de radiação. Os físicos que foram indicados para conferir as máquinas foram condenados à morte. Causa: O software calculava a dosagem de radiação baseando-se na ordem de entrada dos dados, e algumas vezes enviava o dobro da dose do que deveria (Ribeiro, 2012).

2.4 Ciclo de vida de testes de software

Assim como os desenvolvedores seguem o ciclo de vida para desenvolvimento de um software, os testers também acompanham o ciclo de vida dos testes de software, que é nada mais que a sequência de atividades realizadas pela equipa de testes, desde o início do projeto até o entrega do produto.

A estrutura básica do ciclo de vida de testes de software é dividida em 5 etapas fundamentais (Farias, 2014):

- **Planeamento -** elaborar a estratégia de teste, o plano de teste, e a análise de risco do projeto de testes. Esta etapa acontece paralelamente às atividades de levantamento de requisitos do projeto de desenvolvimento do software, e nela devem ser realizados os testes de verificação sobre os requisitos levantados do software.
- **Preparação -** preparar o ambiente de testes para a execução. Entende-se por ambiente de testes os equipamentos, hardware e software, ferramentas para automação de testes, entre outros. Ou seja, toda a estrutura necessária para execução dos testes deverá estar disponível nessa etapa.
- **Especificação -** elaborar e rever os casos de testes e os roteiros de testes. Os casos de testes poderão e deverão ser alterados (inclusão de novos casos), de acordo com a disponibilização por parte da equipa de desenvolvimento dos módulos a serem testados.
- **Execução -** executar os testes conforme planeamento nas etapas anteriores, e registrar os resultados obtidos durante a execução.
- **Entrega -** finalizar o projeto de testes, arquivar a documentação referente ao projeto e relatar todas as ocorrências encontradas nesse projeto a fim de melhorar o processo.

2.5 Níveis de Testes de Software

Os níveis de teste servem para identificar as áreas em falta e evitar sobreposição e repetição entre as fases do ciclo de vida do desenvolvimento. Cada fase de um processo de desenvolvimento de software tem que ser testada, e os níveis de testes são as seguintes (Pinheiro, 2015):

Testes unitários - verificam o funcionamento de módulos de software, programas, objetos ou classes que possam ser testados separadamente. Este tipo de testes é escrito pelos programadores durante o desenvolvimento e consiste no isolamento de partes do programa para evitar dependências, para que haja a certeza que uma determinada função faz o esperado e para que os erros sejam corrigidos logo que forem detetados.

Testes de integração - procuram testar as interações entre diferentes partes de um sistema. O propósito dos testes de integração é fazer uma verificação funcional, de desempenho e de requisitos de confiabilidade dos componentes. Estes testes são feitos com base nos módulos testados nos testes unitários agrupando-os em componentes resultando assim num sistema integrado.

Testes de sistema - testam o comportamento de todo o sistema, para verificar se este cumpre os requisitos especificados. Focam-se nos defeitos que surgem num alto nível de integração. Nestes testes o programa deve funcionar conforme o esperado e o ambiente deve ser idêntico ao ambiente do utilizador final de forma a minimizar o risco de falhas específicas devido ao ambiente e que não foram detetadas em testes anteriores.

Testes de aceitação - Estes testes servem para verificar se o sistema se comporta de acordo com o que foi pedido e determinam a satisfação do cliente relativamente ao produto. São em alguns casos da responsabilidade do utilizador final.

2.6 Tipos de Testes de Software

Existem diferentes tipos de testes de software estando os tipos focados num objetivo particular de cada teste, ou seja, cada tipo de teste depende dos seus objetivos e implica uma organização diferente. Descrevem-se seguidamente os quatro principais tipos de testes de software (Pinheiro, 2015).

Testes funcionais ou testes de caixa-preta - são um tipo de testes baseados nos requisitos funcionais. Nestes testes são verificadas as funcionalidades do sistema sem entrar na sua estrutura interna, apenas é considerado o comportamento externo do software. Este tipo de testes será mais detalhado na secção seguinte por se tratar do tipo de teste enquadrado no âmbito do estágio.

Testes não funcionais - referem-se a aspetos do software que não estão relacionados com nenhuma função, como o desempenho ou a escalabilidade. São os testes necessários para medir as características do sistema, que podem ser quantificadas, como os tempos de resposta nos testes de desempenho.

Testes caixa-branca ou estruturais - avaliam o comportamento interno dos componentes do software. Este tipo de testes atua diretamente no código para avaliar aspetos como condições, ciclos ou caminhos lógicos. Exemplo deste tipo de testes são os testes unitários.

Testes de regressão - são as repetições dos testes a uma aplicação que sofreu alterações de modo a corrigir defeitos para garantir que essas alterações não tiveram uma implicação negativa na aplicação. Caso estas alterações resultem em novos defeitos considera-se que o sistema regrediu. Este tipo de testes inclui testes funcionais, não funcionais e estruturais.

2.7 Testes Funcionais

O teste funcional é um elemento fundamental para assegurar a qualidade de um produto de software, confirmando que este executa suas funções de modo que os utilizadores esperam (Marszałek, 2016).

Nos testes funcionais o tester só sabe o que o software deve fazer – não precisa analisar o código para ver como ele funciona, isto é, se o tester introduzir

uma determinada entrada, ele recebe uma determinada saída, ou seja, não sabe como ou por que isso acontece, apenas o que ele faz (Patton, 2001). O teste funcional envolve dois passos principais: identificar as funções que o software deve realizar e criar casos de teste capazes de verificar se essas funções são realizadas pelo software (Barbosa, et al., 2010).

Os testes funcionais permitem demonstrar, de forma sistemática, que as funções disponibilizadas estão de acordo com o especificado nos requisitos técnicos e requisitos de negócio, na documentação do sistema e nos manuais de utilizador.

A implementação de testes funcionais é bastante utilizada em empresas que seguem a metodologia ágil, onde são realizadas constantes mudanças e introdução de novas funcionalidades.

Este tipo de teste é usado para verificar se a aplicação ou software executa as funções pretendidas através de uma resposta adequada aos comandos do utilizador, verificando a consistência da interface com o utilizador, integração com outros sistemas e processos de negócios. Os testes são realizados com ajuda de recursos de software para determinar o seu comportamento, usando uma combinação de entradas, simulando as condições normais de funcionamento, e anomalias deliberadas e erros (Sommervile, 2011).

Os testes funcionais, normalmente são realizados durante os níveis de teste de sistema, e testes de aceitação, envolvendo seguintes passos:

- Identificar as funções do software.
- Criar dados de entrada com base nas especificações.
- Determinar a saída com base nas especificações.
- Executar casos de testes.
- Comparar os resultados das saídas reais e esperados.

Os testes funcionais são realizados por uma equipa de testes, e requerem algum grau de coordenação entre os membros da equipa. Essas pessoas precisam ser flexíveis para se adaptar às condições de mudança e adotar novos papéis dentro da equipa. O grupo tem a responsabilidade, não só, de projetar e desenvolver testes com base em requisitos específicos, mas também realizar esses testes e, em seguida, analisar e relatar quaisquer defeitos encontrados. A equipa de teste é responsável pelo planeamento de testes com base nos requisitos de software identificadas, e as

atividades de testes são coordenadas com a equipa de desenvolvimento estando muito ligados na procura da melhoria de qualidade.

Os erros encontrados precisam ser explicitados de forma clara para que os defeitos possam ser percebidos e corrigidos em menor tempo possível; os testes funcionais podem ser realizados através de técnica de testes manuais ou testes automáticos.

2.8 Testes Manuais

O teste manual consiste na reprodução de todas as funcionalidades pelo tester seguindo o documento previamente definido. Normalmente quando a interface com o utilizador do software está em causa, um teste manual é muito utilizado, sendo que uma pessoa real pode, mais facilmente, detetar problemas potenciais e reais de um programa em uso, fornecendo acesso rápido aos resultados do teste de uma forma fácil para outros testers e desenvolvedores perceberem o problema.

No contexto ágil, em geral, os testes manuais são vistos, primariamente, como testes exploratórios (Bortoluci & Duduchi, 2015). Executam-se testes através de um ciclo rápido de passos para o planeamento de testes, design e execução.

O teste manual consiste em testar software manualmente, isto é, sem usar qualquer ferramenta automatizada ou qualquer script. Neste tipo, o tester assume o papel de um utilizador final e testa o software para identificar qualquer comportamento ou bug inesperado. Os testes manuais podem ser executados em diferentes níveis, por exemplo teste de unidade, teste de integração, teste de sistema e teste de aceitação por parte do utilizador.

Teste manual, por vezes, não é muito claro, podendo levar muito tempo para a sua execução, não se conseguindo entrar profundamente no funcionamento de muitas aplicações, podendo, às vezes, produzir resultados que estão sujeitos a erro humano.

As atividades de testes manuais são muito repetitivas, a experiencia do tester e o domínio do sistema a ser testado têm grande influencia no sucesso dos testes.

O tester de software precisa de ter visão para testar com sucesso uma aplicação de software, em alguns casos, a visão do tester pode ser mais importante do que o próprio processo de testes (Myers, 2004).

Os testers utilizam planos de teste para realização das tarefas, casos de teste ou cenários de teste, para testar um software garantindo a integridade dos testes.

2.9 Testes automáticos

A automação de testes é uma série de atividades que incluem o desenvolvimento de scripts de teste (código de teste) e execução de scripts de teste, verificação de requisitos de teste e utilização de ferramentas de teste (Collins, 2013). Automatizar testes reduz o esforço das atividades manuais fornecendo respostas rápidas e resultados para a equipa do projeto.

Os testes nunca poderão ser totalmente automáticos, já que os testes automáticos só podem verificar se um programa faz aquilo a que é proposto (Sommervile, 2011). É praticamente impossível usar testes automáticos para testar os sistemas como por exemplo, a interface gráfica de utilizador ou para testar se um programa não tem efeitos colaterais indesejados.

Os testes automáticos ao contrário de testes manuais, fornecem uma solução mais conveniente quando se trata de tipos de desenvolvimento ágil quando são elaborados de forma de adequada. O conhecimento na área de automação e experiência nos métodos e ferramentas, bem como o uso de técnicas que promovem a reutilização e facilidade de manutenção dos testes automáticos, são fundamentais para se obter êxito nesta área (Correia & da Silva, 2004).

Normalmente estes tipos de testes permite a revisão de código automatizado para a cobertura do teste. Entretanto, é importante considerar que este tipo de teste não pode ser usado em todos os casos. É preciso analisar cada projeto para definir quais os casos em que, realmente vale a pena automatizar, e considerar que, mesmo se decidir utilizar é importante ter em conta ferramentas de automação de teste, sendo que ainda poderá ser necessário utilizar testes manuais para diversas funcionalidades.

Este tipo de teste é diferente dos testes manuais, porque requer o conhecimento de boas práticas de programação, conhecimento de ferramentas e arquitetura de software.

2.10 Testes em desenvolvimento Ágil

Testes Ágeis são atividades da etapa de testes num processo de desenvolvimento de software que adotam conceitos de metodologias ágeis. Os métodos ágeis defendem uma equipa de testes mais dinâmica e participativa, logo, é importante que as atividades de testes estejam envolvidas no projeto desde o início e que a equipa participe nas reuniões com o cliente para conseguir perceber melhor quais as suas necessidades (da Silva & Moreno, 2011).

Estimula-se também uma grande interação e colaboração com a equipa de desenvolvedores para alcançar os níveis de qualidade esperados pelo cliente. Iterações curtas são comuns em métodos ágeis, pois assim se permite maior flexibilidade para mudanças (Gregory, 2009). Este contexto ágil exige o apoio das ferramentas para auxiliar na automatização do processo.

Comparando com as metodologias de desenvolvimento tradicionais, como a *Waterfall*, as tarefas de testes não acontecem apenas no final do processo, como as tarefas iterativas e incrementais, as atividades de teste ocorrem ao longo da iteração.

2.10.1 Planeamento

Ao nível do planeamento de testes, considera-se que existem dois tipos, a *release* e a iteração. O planeamento da *release* pode ocorrer meses antes do início do projeto e visa depurar o *product backlog* (lista de todas as funcionalidades desejadas para um produto). Inclui atividades como definição de prioridades, analise de risco, a definição dos níveis de testes e quais as *release*s que vão ser testadas (Cohn, 2004).

O planeamento da iteração acontece depois do planeamento da *release*, de acordo com as prioridades definidas anteriormente. Neste planeamento pretende-se definir as *user stories (*requisitos e elementos que as equipas de programação precisam criar*)* e o trabalho necessário por cada uma delas durante a *sprint*. (Cohn, 2004) . Aqui é feita uma análise mais pormenorizada quanto às funcionalidades que vão ser testadas, tais como o que vai ser testado, como, e qual o esforço necessário para essas tarefas

2.10.2 Quadrante de testes em metodologias ágeis

Os quadrantes dos testes em metodologias ágeis, alinham os níveis de testes com os tipos de testes adequados. O modelo dos quadrantes de testes exibido na ilustração 5 ajuda a garantir que os tipos e níveis de testes importantes são incluídos no processo de desenvolvimento.

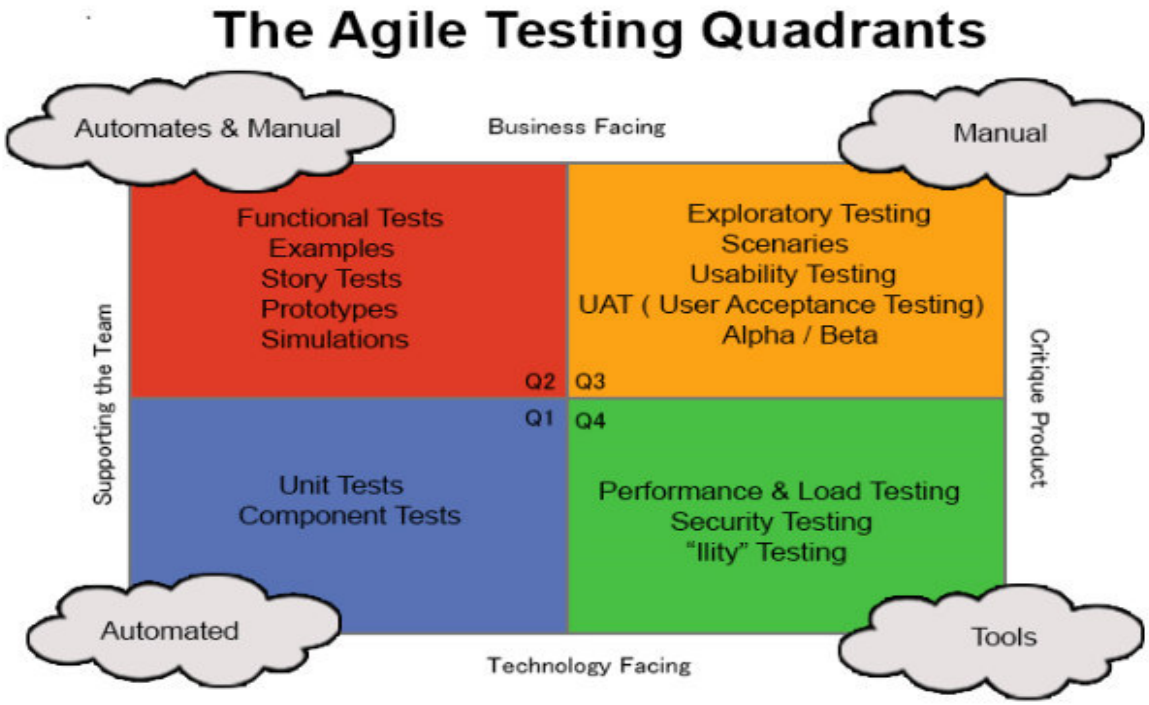

Ilustração 5 - Quadrantes de testes das metodologias ágeis (Shiralige, 2016)

A ordem de numeração dos quadrantes não está relacionada com o momento em que cada teste deve ser feito. O momento para cada tipo de teste está relacionado com os objetivos e riscos associados ao projeto. Nas linhas abaixo serão explicados cada um dos quadrantes (Gregory, 2009).

Quadrante 1

- Q1, representa os testes num nível mais baixo.
- Incluem-se os testes unitários e os testes de componentes.
- O objetivo principal deste é orientar os programadores durante o desenvolvimento, permitindo-lhes maior confiança na escrita do código.

Quadrante 2

- Q2, também ajudam o trabalho da equipa de desenvolvimento, embora que num nível mais alto.
- Representam a qualidade desejada pelos clientes, por isso se chamam testes voltados para o negócio ou para o cliente.
- Derivam de exemplos debatidos com o cliente que descrevem cada funcionalidade.
- São escritos numa linguagem que possa ser entendida pela área de negócio.
- A maior parte destes testes necessitam de ser automatizados, de forma a fornecer informação de forma rápida e para que possam ser executados com frequência como parte de um processo de integração contínua.

Quadrante 3

- Q3, são utilizados para criticar o produto.
- Identificam se o software desenvolvido está dentro das expectativas ou não.
- Visam simular o comportamento do utilizador final.
- Em certos casos estes testes podem ser automatizados, mas noutros apenas a intervenção humana poderá ditar a qualidade do produto.

Quadrante 4

- Q4, é transversal a qualquer metodologia de desenvolvimento.
- Tem como objetivo analisar questões de desempenho, robustez e segurança, através de ferramentas especializadas.

2.11 Ferramentas de automatização de testes

As primeiras ferramentas de testes de software surgiram por volta de 1980 e as suas principais funcionalidades baseavam-se no *Debuging* do software. Após essas ferramentas, surgiram as ferramentas de gestão de testes que tinham como objetivo organizar e guardar dados dos testes, organizar o resultado da execução dos testes e apresentar relatórios de testes. Por volta de 1995 surgiram as primeiras ferramentas para automatização de casos de testes. Estas eram inicialmente focadas em medições básicas de algumas plataformas (Carvalho, 2010).

Atualmente as ferramentas de automatização são mais completas e são de mais fácil utilização permitindo a realização de um número maior de funções em relação às ferramentas desenvolvidas anteriormente.

Este facto, deve-se à automatização de testes de software ser vista como uma das principais medidas para melhorar a eficiência da atividade de teste, agregar qualidade ao produto e antecipar a descoberta de falhas e incompatibilidades, reduzindo assim o custo do projeto e melhorando a produtividade a vários níveis (Carvalho, 2010).

Existem um grande número de ferramentas disponíveis no mercado, apenas serão referidos alguns exemplos seguintes:

- **Selenium** – ferramenta *open source* para automatizar a execução de testes em sistemas web. O selenium IDE é um Add-on para o navegador Firefox que permite a gravação de um roteiro durante a execução de um teste manualmente.

- **IBM Rational Functional Tester (RFT)** é uma ferramenta de teste funcional e de regressão automatizada para equipas de garantia de qualidade.

- **HP Quick Test Professional -** conhecido pelo UFT (Unified Functional Testing), é uma ferramenta de automação para casos de teste funcionais e de regressão. Permite testar as mesmas ações para diferentes utilizadores, conjunto de dados, diferentes versões do sistema operativo Windows e diferentes navegadores web.

- **Watir** - ferramenta de testes *open source*, utilizada para automatizar o teste de aplicações executadas num browser através da escrita de scripts de teste de fácil interpretação e manutenção. Permite executar o browser como um utilizador, como por exemplo selecionar links, preencher formulários, pressionar botões é depois verifica os resultados.

- **Sahi** - ferramenta de testes open source, a criação de scripts de teste é feita através de um servidor Proxy posicionado entre o browser e o servidor da aplicação alvo de teste.

2.12 Limitações dos testes de software

Apesar dos testes de software serem essências no processo de desenvolvimento de um software a sua implementação apresenta algumas limitações ou exigências importantes. Uma das mais importantes limitações dos testes de software é que o teste pode apenas mostrar a presença de falhas, e não a sua ausência (Ammann & Offutt, 2008).

Assim sendo, testar não permite garantir que um software funciona corretamente em todas as condições, mas apenas garante que o software não funciona para uma determinada condição.

As empresas pretendem testar tudo antes da entrega do software ao cliente. Mas esse "tudo" é bastante ilusivo e pode referir-se a um dos vários pontos mencionados a seguir (Singh, 2011):

- Executar todas as ações do programa;
- Executar todas as condições verdadeiras e falsas;
- Executar todas as condições de decisão;
- Executar todos os caminhos possíveis;
- Executar com todo o input de dados válidos;
- Executar com todo o input de dados inválidos.

É impossível alcançar todos estes pontos, ou até mesmo um único ponto, devido às limitações do tempo e recursos disponíveis nas várias fases dos projetos. Atualmente desenvolvem-se softwares para diferentes sistemas operativos, o que torna ainda mais difícil os testes, assegurando a estabilidade do produto em diferentes sistemas ou dispositivos. A execuções de testes não prova totalmente, que um software está habilitado funcionalmente para entrar em produção. Testar exaustivamente é impraticável, ou seja, testar para todos os possíveis elementos de entrada e saída (Myers, 2004). A implementação de atividades de testes de software é cara e trabalhosa, podendo consumir grande parte dos custos de desenvolvimento de software (Ammann & Offutt, 2008).

2.13 Testes a aplicações web e aplicações mobile

As aplicações web são executados na internet, ou seja, os dados ou os arquivos são processados e armazenados na web. Estas aplicações, por norma, não necessitam de uma instalação prévia no computador do utilizador final, qualquer utilizador pode usar acedendo ao endereço URL dessa aplicação. Toda a informação é guardada de forma permanente em servidores de internet, e estes enviam perante um dado input uma resposta aos dados requeridos. Em qualquer momento, lugar e em qualquer dispositivo podem ser acedidos os serviços dessas aplicações, sendo apenas necessária uma ligação à internet.

A área da aplicação mobile é uma área em crescimento nos últimos tempos, tendo surgido de uma forma natural devido à grande evolução dos *tablets* e *smartphones* e também à crescente popularidade dos dispositivos móveis (Anacleto, 2012). As aplicações mobiles são desenhadas para os seguintes sistemas operativos:

- Android da Google
- IOS da Apple
- Windows Phone da Microsoft (wp).

Uma realidade muito comum, é que as empresas detentoras de determinadas aplicações web têm vindo a apostar no desenvolvimento da aplicação mobile, por forma a tornar a utilização mais cómoda para o utilizador.

As aplicações mobile podem ser consideradas semelhante às aplicações web, mas, não devem ser vistas como uma versão para dispositivos móveis de sites. Por exemplo, o facebook tem um site na internet, existindo versões *responsive,* ou seja, têm um comportamento adaptativo a qualquer dispositivo móvel, mas, no entanto, disponibiliza o aplicativo que pode ser instalado no próprio dispositivo, quer seja num sistema operativo Android ou mesmo iOS.

Nem todas as aplicações funcionam da mesma forma, alem de que as aplicações web e mobile são destinados a dispositivos diferentes; o tester no momento dos testes deve ter em conta as seguintes características:

Numa aplicação web:

- O browser
- O responsive

Numa aplicação Mobile:

- Sistema Operativo (iOS, Android ou wp);
- Versão do sistema operativo;
- Tipo de dispositivo (Telemóvel/Tablet/tamanho do ecrã).

Estas características que parecem não oferecer qualquer obstáculo, originam na fase de testes erros mais comuns das divergências de funcionamento das aplicações nos dispositivos diferentes. Para clarificar melhor a ideia das características, por exemplo, uma aplicação num dispositivo *Android* versão 4.3 funciona bem e na outra versão 4.0 poderá não funcionar corretamente.

Outro exemplo tem a ver com o tamanho dos dispositivos, mais precisamente nos tablets que, normalmente existem em 3 tamanhos, muitas vezes o que acontece é a desformatação da aplicação num Tablet de tamanho inferior em comparação com outro de tamanho superior.

Relativamente a testes de aplicações web e aplicações mobile, pode-se dizer que é muito mais fácil testar uma aplicação web do que uma aplicação mobile, visto que as principais caraterísticas que se devem ter em conta no teste de uma aplicação web é o tipo do browser (acontecendo situações como a desformatação ou não funcionamento num determinado browser) e aplicação do *design* tipo *responsive*.

O teste de software para dispositivos móveis, apresenta desafios próprios devido às aplicações móveis serem cada vez mais complexas, o que faz com que as atividades de teste sejam cada vez mais fundamentais, podendo uma falha nessas atividades significar um grande impacto económico (Coelho, 2016).

Numa aplicação mobile os testes são mais trabalhosos e demorados visto que a aplicação tem de ser testada em vários dispositivos diferentes, aumentando o trabalho também para a equipa de desenvolvimento. Normalmente desenvolve-se versões para os três sistemas operativos diferentes o que ainda dificulta os testes.

3. Testes na ITsector

Neste capitulo descrevem-se as tarefas realizadas durante o estágio, de acordo com métodos de trabalho da empresa. Todas as informações, processos e definições neste capitulo têm como fonte um manual interno da metodologia de desenvolvimento de software da Itsector (Itsector, 2012).

O projeto de estágio foi desenvolvido na ITsector que se encontra equipada com equipamentos e tecnologias necessários para realização do estágio. O excelente processo de integração foi fator chave na adaptação ao ambiente de trabalho, bem como passar conhecimentos relevantes para dar respostas aos objetivos pedidos. Todo o processo de estágio foi orientado para um melhor enquadramento às metodologias utilizadas na empresa na área dos testes, bem como uma orientação ao longo da realização de tarefas atribuídas no estágio.

As tarefas atribuídas ao longo do estágio estavam enquadradas dentro das tarefas da equipa de testes da Itsector. Os projetos desenvolvidos na Itsector têm etapas bem definidas durante o ciclo de vida do mesmo, desde períodos de análise/ levantamento de requisitos, período de desenvolvimento e testes.

No momento da atribuição de um projeto, foram passados os documentos funcionais dos mesmos, para que, desde o inicio, se compreendessem os requisitos, quais as ações e os resultados esperados das diversas funcionalidades do produto em questão. Perante isto, eram dadas as condições necessárias para que o tester desenhasse os casos de teste para cada funcionalidade. Deste modo após o desenvolvimento dos testes de integração por parte do programador, bem como a passagem do produto para um ambiente adequado para os testes, o tester, com base nos casos de teste especificados, tem condições para iniciar as suas tarefas no âmbito do projeto, sendo que será o tester a dar o feedback de como se encontra a aplicação antes da disponibilização para o cliente final.

Os testes funcionais são, por norma, testados em ambiente de qualidade, no entanto, poderia por vezes ser necessário testar em ambiente de desenvolvimento para muitas vezes auxiliar o programador a perceber qual é, de facto, o problema.

Em síntese, as aplicações são testadas em ambiente de qualidade antes de irem para o cliente final para que este faça todos os testes de aceitação necessários. Quando entregue ao cliente, caso haja bugs, o cliente reporta à Itsector, os programadores corrigem, é feita uma verificação pelos testers da empresa, por forma

a garantir que realmente o problema foi corrigido e posteriormente, após uma entrega ao cliente é oficializado que a versão entregue possuí as correções pedidas.

Os clientes da ITsector têm apostado no desenvolvimento da aplicação mobile, ou seja, a maioria dos projetos são desenvolvidos em duas versões (web e mobile), isto aumenta as dificuldades na realização das tarefas de testes consumindo mais tempo.

A aplicação, depois da aceitação por parte do cliente, é passada para o ambiente de produção, muitas vezes de um modo piloto, quer isto dizer que apenas um grupo restrito de pessoas é que tem acesso à aplicação em produção.

Estas fases são muito importantes, na medida em que neste sistema "mais real" poderão ser encontrados problemas que antes não foram detetados. É, pois, notório que as realizações de testes fazem parte de um processo contínuo e ativo na fase de vida de um projeto.

3.1 Metodologia de desenvolvimento de software da Itsector

Antes de iniciar com a fase de testes é necessário um enquadramento com a metodologia de desenvolvimento da empresa. A qualidade da implementação da solução proposta será controlada e assegurada por uma metodologia desenvolvida pela ITsector que tem como objetivo, não só, uma implementação de qualidade, mas também o desenvolvimento do produto no que diz respeito ao processo inerente à sua concepção.

Ao nível do processo a qualidade mede-se pela observância de determinada metodologia ou ciclo de vida de desenvolvimento, bem como pela correta integração dos subprodutos gerados e transmitidos entre as sucessivas fases.

A metodologia de desenvolvimento da ITSector é composta pelas seguintes fases apresentadas na figura 6:

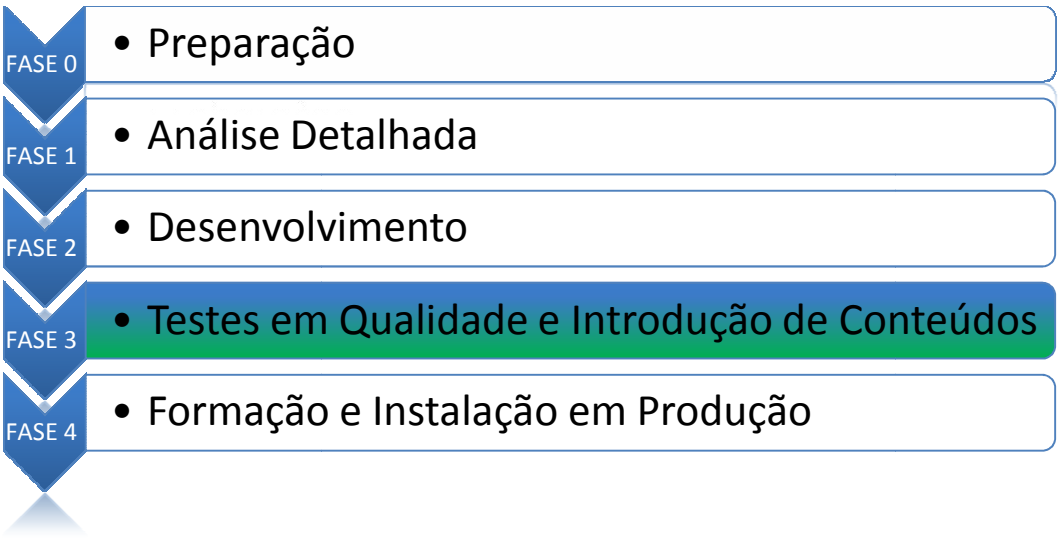

Ilustração 6 - Metodologia de desenvolvimento Itsector

Cada fase é caracterizada por um conjunto de objetivos a atingir, pontos de controlo e deliverables produzidos, tal como seguidamente se descreve.

3.1.1 Fase 0 - Preparação

Esta etapa consiste na preparação do Kick-Off do projeto, caracteriza-se por:

- Definição do âmbito do projeto
- Definição de objetivos
- Definição da estrutura organizacional do projeto
- Avaliação das condições logísticas
- Planeamento de ações
- Preparação da reunião de arranque do projeto

3.1.2 Fase 1 – Análise detalhada

Esta etapa pode ser dividida em duas partes: uma para a identificação e formalização de todos os requisitos e outra para o desenho global da solução com base nos requisitos aprovados. No final desta fase deverá estar disponível toda a documentação necessária para o início da implementação.

Esta etapa caracteriza-se por:

- Reuniões de levantamento de requisitos funcionais
- Análise dos processos de negócio
- Elaboração de documentação preliminar
- Validação da especificação dos processos
- Elaboração de documento de especificação funcional
- Elaboração de documento de especificação técnica

3.1.3 Fase 2 – Desenvolvimento

Inicia-se a fase de implementação do projeto. São programados os processos definidos, mecanismos de integração de dados, estrutura de navegação e menus da aplicação, paginas web, reports, etc. São ainda parametrizadas as variáveis dinâmicas, bases de dados entre outros requisitos definidos na análise.

Esta etapa caracteriza-se por:

- Concepção e desenho gráfico
- Desenvolvimento de protótipo gráfico
- Definição e desenvolvimento de layouts
- Definição e desenvolvimento de interfaces
- Desenvolvimento da solução
- Validação e aceitação de processos

3.1.4 Fase 3 – Testes em qualidade e introdução de conteúdos

Esta fase foca-se fundamentalmente na garantia da qualidade da solução, sendo testadas as diversas funcionalidades, bem como na garantia da conformidade da solução implementada com os requisitos iniciais. O objetivo dos testes é definir o que vai ajudar e/ou impedir os utilizadores de cumprirem eficazmente os propósitos

da solução implementada. Os responsáveis pelos testes devem construir um cenário de funções a validar.

Esta etapa caracteriza-se por:

- Desenvolver o plano de testes;
- Pedir dados para a realização dos testes;
- Desenvolvimento de casos de teste (Criar Lista de Testes de Integração, definir casos de teste, identificar dados de teste, sequenciar a execução dos casos de teste, atualizar lista de testes de integração)
- Preparar ambiente de testes
- Execução de testes
- Gerir o ambiente de testes;
- Análise dos resultados de testes
- Criação de relatórios para a gestão
- Aprovar testes de aceitação.
- Entregar os módulos para testes dos utilizadores previamente à implementação em piloto
- Corrigir erros detetados nos testes e implementação de melhorias;
- Aceitação provisória do projeto;
- Desenvolvimento de alterações;

3.1.5 Fase 4 – Formação e instalação em produção

Após concluídas todas as baterias de testes e existindo uma pré-aprovação de todos os requisitos pelo cliente, deverá ser efetuada a formação aos utilizadores finais, ao mesmo tempo que será planeado o arranque da solução em produção. Após o arranque da solução, e se aplicável, inicia-se um período de acompanhamento em produtivo.

Esta etapa caracteriza-se por:

- Preparação das infraestruturas de produção
- Formação aos utilizadores

3.2 Princípios de testes na Itsector

Para que a aceitação em ambiente de qualidade decorra normalmente deve-se alcançar, no projeto desenvolvido, níveis de qualidade que sejam ótimos para os utilizadores; desta forma a fase de testes é essencial e devem respeitar as seguintes características principais:

- Para serem eficazes os testes devem ser cuidadosamente desenhados e planeados – para isso a Itsector entrega um plano de testes global para cada projeto.
- Testes improvisados devem ser evitados.
- Durante e após a realização dos testes, os resultados de cada teste devem ser minuciosamente inspecionados, comparando-se resultados previstos e obtidos.
- Os programadores não são as pessoas mais adequadas para testar os seus próprios produtos.
- O trabalho dos *testers* no controlo de qualidade do projeto é muito importante nos testes e na aceitação final por parte do cliente.
- O planeamento dos testes e respetivo manual de testes será elaborado por elementos experientes que conhecem profundamente a metodologia adotada.
- A realização dos testes baseados em planos e documentos com especificações bem estruturadas pode ser desempenhado por elementos menos experientes.

3.3 Auditorias de Qualidade

A fase de teste é o processo essencial para garantir a qualidade de software visto que apesar de existirem excelentes processos de desenvolvimento de software, por si só não são capazes de garantir a qualidade do software. Os defeitos de software podem ser introduzidos durante as diversas fases do ciclo de vida de desenvolvimento de software.

Os objetivos dos testes são de encontrar defeitos e verificar que o software responde aos seus requisitos de qualidade, incluindo atributos, como entendido pelo utilizador. Infelizmente, em muitos casos, os ensaios são na realidade programas que visam apenas validação da funcionalidade. Isso fica aquém dos objetivos de um verdadeiro teste. Testes que não abrangem os atributos de qualidade geralmente não são considerados completos.

A ITsector inclui auditorias de qualidade de software em programas para fornecer uma garantia razoável de qualidade. Todos os atributos de qualidade especificados para os requisitos do software deverão ser testados. Só através de uma auditoria de teste se pode ganhar um certo nível de confiança na qualidade do software em análise.

A qualidade não acontece por acaso. O processo deve ser gerido. Todas as pessoas envolvidas no desenvolvimento do projeto de software devem ser consideradas responsáveis por satisfazer os seus requisitos de atributos de qualidade. O processo também deve ser comunicado e compreendido.

Auditorias de qualidade são realizadas não só para encontrar defeitos de software ou questões de não conformidade, mas também para promover a qualidade de software no entendimento da equipa de desenvolvimento. Antes da auditoria de qualidade ou ensaio, a abordagem da equipa de auditoria e as técnicas que irão utilizar, devem ser comunicadas à gestão, arquitetos e programadores, de modo a que todos compreendam o que a auditoria irá dar origem e a forma como poderão ser detetadas deficiências.

Além de ajudar o processo de auditoria de forma eficaz, este também oferece uma oportunidade para que todos possam pensar em formas de melhorar a qualidade em todo o desenvolvimento.

3.4 Principais atividades de testes

Para atingir os objetivos dos testes, a ITsector, com base numa vasta experiência de implementação e testes de aplicações, considera necessárias as seguintes atividades como apresenta a figura 7:

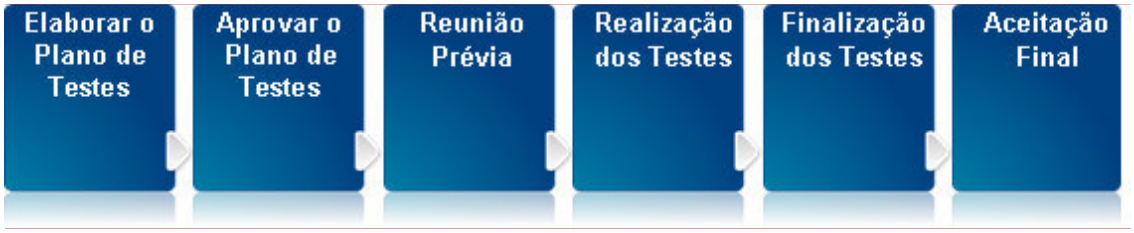

Ilustração 7 - Atividades de Testes na Itsector

Seguidamente serão descritas da forma mais pormenorizadas cada uma das atividades:

3.4.1 Elaborar o Plano de Testes

Será elaborado um plano de testes da aplicação, o qual:

- Estabelece a estratégia de condução dos testes;
- Define os critérios de aceitação;
- Define o ambiente e a configuração de *hardware* e *software* requeridos para os testes;
- Estima os recursos humanos e materiais necessários, as atividades específicas e a sua distribuição no tempo;
- Estas atividades devem cobrir os aspetos de preparação (definição de casos de teste e resultados esperados), execução, análise e relatórios de resultados.

3.4.2 Aprovar o Plano de Testes

- A ITsector desenvolverá em conjunto com a o cliente o plano de testes associado ao sistema.
- O cliente deverá remeter por escrito a aprovação ou as alterações consideradas oportunas.

3.4.3 Reunião Prévia

Terá lugar uma reunião antes do início dos testes. Nesta reunião deverão participar: o Coordenador de Projeto do cliente, o Coordenador de Projeto da ITSector e a equipa de testes, com a finalidade de:

- Concretizar e atualizar o calendário previsto;
- Apresentar a configuração real sobre a qual serão executados os testes;
- Apresentar o grupo encarregue de executar os testes;
- Comentar o plano de testes e os casos de teste adicionados pelo cliente;
- Estabelecer a sequência de atividades mais indicadas.

3.4.4 Realização dos Testes

A realização dos testes ocorrerá após a entrega do software ou aplicação para teste e respeitará os seguintes pontos:

- Será respeitado o planeamento estabelecido no Plano de Testes;
- Apenas serão considerados os testes patentes no Plano de Testes;
- Repetir-se-ão os casos de teste cujo resultado não tenha ficado claramente definido como positivo ou negativo;
- À medida que se forem executando os casos de teste serão registados os resultados e será confirmado se vão de encontro com o esperado. Caso isso não se verifique, serão anotados como problemas, discrepâncias ou pontos em aberto, consoante o caso;
- Caso se produza algum destes incidentes (problema, discrepância ou ponto em aberto), serão compilados todos os dados possíveis, incluindo as circunstâncias e a configuração sobre as quais ocorreu o incidente, para facilitar a análise posterior;
- Se a solução para um dado incidente for considerada trivial, poderá implementar-se no ato de teste, sendo este repetido. Caso contrário a

análise do incidente será efetuada depois dos restantes testes da bateria de testes;

- No final da bateria de testes efetuar-se-á uma recompilação de todos os dados obtidos e será feita uma avaliação dos resultados.

3.4.5 Finalização dos Testes

Depois de terminados os testes, a equipa de testes de aceitação elaborará um relatório de testes, no qual estarão patentes os respetivos resultados, que posteriormente serão entregues à ITSector. Todos os incidentes serão classificados como problemas, discrepâncias ou pontos em aberto e será realizada uma análise dos mesmos, consoante a natureza do incidente, visando a sua resolução.

Depois desta análise serão implementadas as ações corretoras necessárias e será planeada a execução de testes de regressão, de forma a garantir que as correções realizadas eliminaram a fonte do problema, discrepância ou ponto em aberto. O procedimento da execução dos testes de regressão será o mesmo que se estabeleceu anteriormente, reduzido ao âmbito dos casos de teste onde se detetou o problema, discrepância ou ponto em aberto.

No caso de terem lugar testes de regressão, será preenchido novo relatório de testes, reduzido ao âmbito destes casos de teste, no qual estarão patentes os respetivos resultados.

3.4.6 Aceitação Final

Uma vez entregues todos os relatórios de teste, indicando o sucesso de todos os testes e a ausência de não conformidades, o cliente declarará formalmente a solução como pronta para se iniciarem as tarefas relativas à formação.

3.5 Ferramenta de apoio (TFS)

A Team Foundation Server(TFS) (https://www.visualstudio.com/tfs/) é uma plataforma de colaboração para projetos de desenvolvimento de software que permite integração entre todos os participantes do projeto, permitindo a gestão de projetos e gestão do código fonte.

A TFS é utilizada para planeamento de trabalho na Itsector, sendo de referir que esta é utilizada pela equipa de gestão de projeto, pelos programadores e a equipa de testes. Esta plataforma é utilizada para planear e gerir seguintes situações:

- Projeto, adicionando item de trabalho para cada user story ou requisito (Backlog)
- Equipa ou equipas (membros e sua disponibilidade)
- O esforço necessário (tarefas do projeto e seu esforço)
- Prazos, dividindo trabalho em iterações
- Testes (reportar bugs, re-testar bugs e correr/criar casos de testes)

Todos os projetos desenvolvidos pela ITsector estão implementados nesta ferramenta onde os intervenientes do projeto interagem, sendo uma ferramenta essencial devido à sua integração com o Visual Studio possibilitando a visibilidade do código fonte das aplicações em desenvolvimento. Sendo a metodologia de desenvolvimento ágil de software um conceito baseado em valores e princípios ágeis, valorizando indivíduos, interações e colaboração do cliente, a TFS ajuda a promover esses princípios por ser uma ferramenta extremamente interativa que facilita o envolvimento de todas as partes envolvidas no projeto.

A TFS é uma ferramenta de fácil de utilização, onde estão alocados todos os projetos e utilizadores da empresa. É composta por muitas funcionalidades, mas para a equipa de teste normalmente é usada para elaboração de casos de testes, reporte de bugs, criação de tarefas e validações das mesmas.

3.6 Qual é o papel do tester?

A principal função do tester é encontrar erros, falhas ou defeitos numa aplicação com intuito de ajudar na otimização da qualidade do produto.

Antes da realização dos testes foi apresentado ao estagiário um documento escrito com objetivo de incutir valores essências para o sucesso de um tester que consistem no seguinte:

- **Criatividade** - a criatividade no momento da execução de testes é muito importante de tal forma que pode ter impacto no sucesso e no tempo da realização dos testes.
- **Noções de programação** - para um tester, apesar de não utilizar o código da aplicação, é uma mais-valia ter noções de programação facilitando na identificação dos problemas.
- **Ser perfecionista** - o tester perfecionista testa ao limite e preocupa-se com pequenos detalhes para que a aplicação esteja a funcionar na perfeição.
- **Objetivo** - a objetividade no momento da descrição dos casos de testes e reporte dos bugs é essencial para um rápido entendimento por parte da equipa de desenvolvimento.

A forma e as estratégias de testes têm muito a ver com a experiencia adquirida e com os projetos. Normalmente na ITsector desenvolvem-se muitos projetos para a área financeira; com o passar de tempo o tester vai-se apercebendo que a base de testes é a mesma podendo, através da experiencia, utilizar estratégias semelhantes.

O objetivo do tester não é mudar a aplicação nem alterar, contudo o tester pode sugerir melhorias ou alterações que, podem, ou não, ser validadas por outras equipas envolvidas no projeto. Muitas vezes a opinião do *tester* difere da opinião da equipa de desenvolvimento, mas, espera-se, para bem do produto, que haja sempre um entendimento na aceitação e resolução dos problemas.

O tester deve ter o cuidado de não reportar bugs que não existem, os bugs devem ser sempre baseados em factos e provas que realmente existem.

3.7 Enquadramento do estagiário na metodologia da empresa

A primeira fase do estágio iniciou-se com ações para enquadramento do estagiário nas metodologias de trabalho da ITsector tendo sido o primeiro passo a instalação da ferramenta TFS no computador do estagiário.

Depois da instalação foi disponibilizado um documento designado de Academia TFS, que consistia numa formação sobre a ferramenta com o objetivo de explicar os principais conceitos. O documento baseava-se num exemplo prático da utilização do TFS no planeamento de um projeto referente a uma aplicação internet banking, apresentando qual seria o enquadramento da equipa de testes.

No que diz respeito a testes estavam descritos os passos e a forma como deveriam ser reportados os bugs ou tarefas. Na prática o projeto era composto por prints de uma simulação de todos os passos realizados durante os testes e a forma como as nomenclaturas deveriam ser usadas para facilitar a comunicação entre as partes envolvidas no projeto. Nesta primeira fase o estagiário teve 15 dias para estudar e adotar as metodologias.

Nem todos os processos de aprendizagem foram adquiridos através da documentação; à medida que o surgiam dúvidas o estagiário realizava as suas próprias pesquisas, e colocava questões á responsável da equipa de testes.

O processo de enquadramento ocorreu dentro da normalidade, ficando o estagiário com o conhecimento das metodologias de trabalho da empresa e noção da organização dos projetos usando a ferramenta TFS.

3.8 Tarefas

Após a segunda semana de enquadramento do estagiário seguiu-se o primeiro projeto, que se tratava de um cliente do sector financeiro. O projeto do cliente bancário consistia numa aplicação Internet Banking em desenvolvimento, com uma versão web e outra versão mobile. A principal atividade nesta fase é o testar ou seja, tendo uma aplicação disponibilizada, procurar erros seguindo um documento de análise onde estão descritas todas as funcionalidades e comportamentos esperados do projeto, servindo como documento de apoio aos testes.

Para realização de testes existe um conjunto de tarefas que são comuns a todos os projetos:

3.8.1 Casos de Testes

Um caso de teste responde à pergunta: "O que testar?" para definir o que deve ser validado para garantir que a aplicação está a funcionar corretamente. Casos de

testes são constituídos a partir de um conjunto de dados de entrada necessários para a execução do software e os respetivos resultados esperados. Representam um cenário de teste, uma situação diferenciada e única de comportamento no software com objetivo de identificar defeitos que ainda não foram descobertos.

Quanto maior for o número de casos de testes maior é a probabilidade de encontrar defeitos no sistema. À medida que vai aumentar o numero de casos de testes, aumenta a confiança na qualidade do produto, visto que cada caso de teste reflete um cenário.

Na Itsector a criação dos casos de testes é definido pela equipa de testes que é responsável pela criação dos nomes de todas as funcionalidades que devem ser testadas. O tester inicia os casos de testes utilizando um documento de análise que serve como referencia; utilizando a aplicação em teste, testa-se cada funcionalidade para verificar se o comportamento obtido é igual ao comportamento esperado.

Para a realização de casos de testes são descritos todos os passos de uma funcionalidade. Por exemplo, para realizar o teste à funcionalidade login, existiriam os seguintes passos:

- Testar login sucesso; caso o login ocorrer com sucesso espera-se que a pagina direciona ao menu principal;
- Caso login não tiver sucesso ou for obtido outro comportamento, o tester regista a falha no teste e descreve qual foi o comportamento obtido comparando com o comportamento esperado.

Continuando com exemplo da validação do login, utilizando a ferramenta TFS o tester para registar o resultado dos testes poderá escolher os seguintes botões:

- **Pass Test** - no caso de sucesso do login;
- **Fail Test** - no caso login ter falhado;
- **Pause Test** - serve para informar que o teste está em pausa;
- **Block Test** - indica que a funcionalidade está bloqueada**;**
- **Not Aplicable** - indica que não conseguiu aplicar o teste.

Esta forma de testar simula a descrição de um caso de uso onde o estagiário faz o teste à aplicação verificando cada resultado de saída, ou seja, se o que está escrito é o resultado que se espera que o sistema faça se não acontecer o esperado, utilizando a ferramenta TFS o estagiário indica a falha e o passo falhado.

Caso de testes é uma forma de teste que leva mais tempo por causa da descrição dos detalhes, mas é uma das melhores formas de realizar testes funcionais. É uma das atividades mais abrangentes e utilizadas pelos *testers*.

A figura 8 mostra um exemplo de um caso de teste realizado de uma aplicação desenvolvida na Itsector, recorrendo ao TFS.

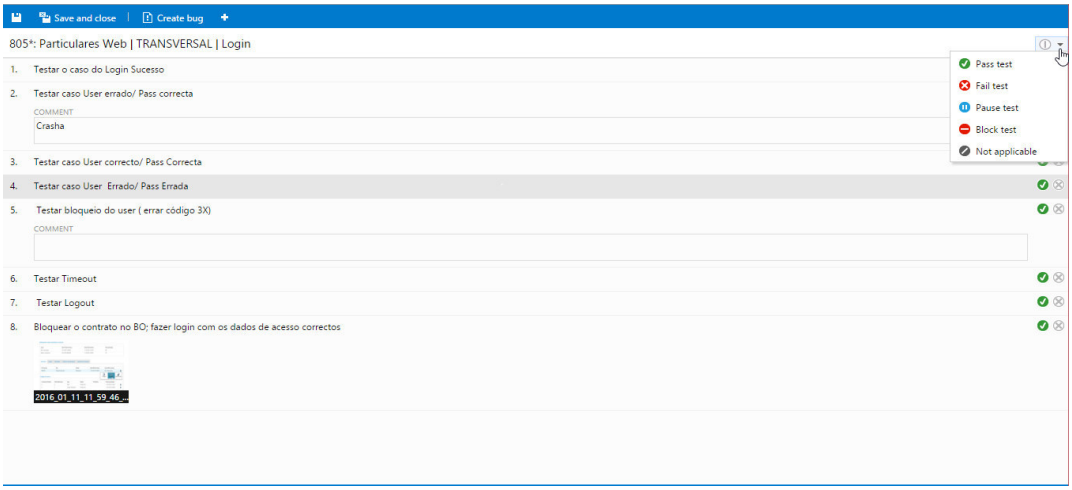

Ilustração 8 - Casos de Teste no TFS

3.8.2 Reporte de Bugs na ITsector

Antes de iniciar o registo do bug é preciso conhecer a nomenclatura utilizada na empresa para criação de um bug, para que este seja mais facilmente percetível para o programador; é necessário descrevê-lo o mais simples e claro possível para que o programador tenha uma maior probabilidade de perceber o problema, ou seja, é fundamental o uso de uma linguagem comum entre o tester e o programador, linguagem esta objetiva e sintética. A nomenclatura utilizada na empresa por todos os testers e programadores para a criação de um bug, utilizando o TFS, é a seguinte:

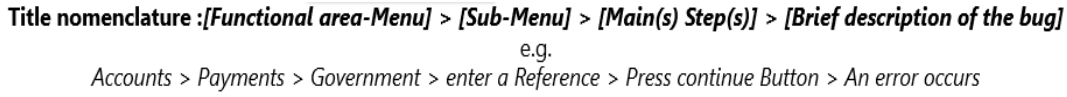

Figura 1 - Nomenclatura de bug na Itsector

A utilização desta nomenclatura permite que o programador, muitas vezes nem precise de ler a descrição ou detalhes do bug criado, ou seja, só pelo título do bug consegue facilmente perceber o problema e, às vezes, em pouco tempo já tem a solução. Por outro lado, um bug mal reportado ou uma nomenclatura diferente leva, muitas vezes, a que o programador tenha dificuldades em perceber o problema, fazendo com que este perca mais tempo a perceber e a resolver o bug.

Normalmente com o decorrer dos testes, o *tester* acaba por ganhar experiências no que diz respeito às formas de procurar e comunicar os bugs. Por isso a experiência do tester é muito importante na procura e registo dos bugs.

3.8.3 Estado de um Bug

Na ferramenta TFS quando um bug é criado começa em estado **New,** e é assinalado para um programador de forma a validar a existência do mesmo. Isto é, o programador utiliza a aplicação e tenta reproduzir o bug descrito pelo tester para confirmar a ocorrência do bug.

Caso o programador conseguir reproduzir o bug descrito, o programador fixa no bug, ou seja, concentra-se no bug para a sua resolução, mudando ao estado do bug para **Active,** isto é, nesta fase já validou a existência do bug e está a tentar resolver o bug. Quando o programador resolve o bug, altera o estado do bug para **Resolved.**

O tester recebe a notificação via email que o bug que reportou já se encontra resolvido. O tester vai à aplicação volta a testar o bug para verificar se realmente está corrigido. Caso o mesmo esteja corrigido altera-o para estado **Closed,** ou seja, confirma a resolução do bug, se não estiver corrigido, altera para estado **Active**, e o "bug" volta para o programador e o ciclo repete até que o bug esteja corrigido.

Existem casos em que um bug que já estava resolvido ou em estado **Closed** volta a ser diagnosticado e é reaberto pelo tester passando-o para o estado **Active** chamando-se de regressão.

São muito comuns casos em que a resolução de um bug origina outros novos bugs, neste caso o tester deverá validar e fechar o bug em resolução e abrir um novo bug.

A figura 9 representa os estados possíveis de um bug, de acordo com o TFS.

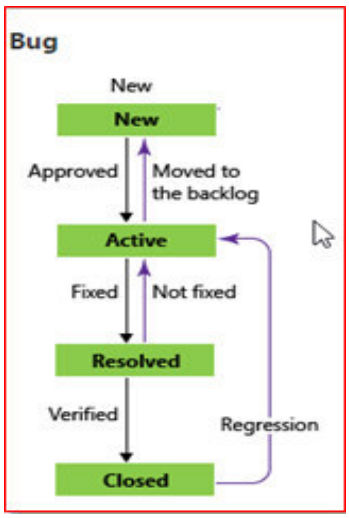

Ilustração 9 - Estado de um Bug

3.8.4 Reportar Bug

O reporte de bug é uma das tarefas mais realizadas pelo estagiário. Consiste na utilização exaustiva da aplicação ou software desenvolvido com o objetivo de encontrar falhas e ajudar na melhoria contínua da qualidade do produto.

O tester inicia os testes da aplicação seguindo o documento de análise, simulando todas as funcionalidades possíveis na mesma, registando qualquer comportamento não esperado como um bug ou falhas detetadas.

Reportar um bug não é uma tarefa difícil, mas exige alguns cuidados a ter na hora de escolher a iteração adequada, para que a *user story* escolhida seja "deslocada" para a iteração atual de modo a ser possível a sua visualização e correção no momento adequado.

Neste exemplo foi escolhido um projeto em desenvolvimento para exemplicar o registo de um bug como apresenta a figura 10.

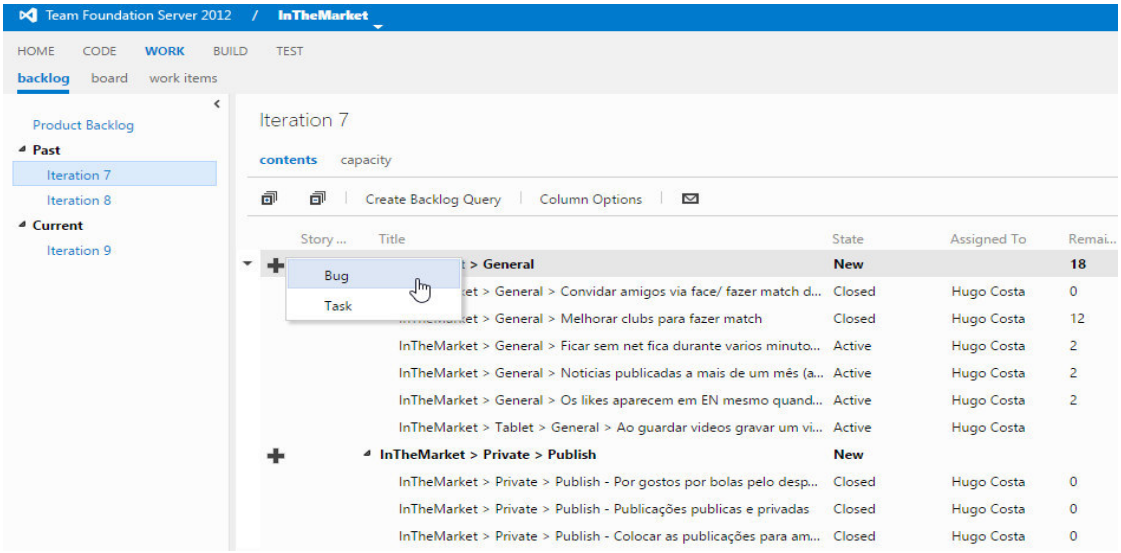

Ilustração 10 - Bug no TFS

Ao escolher a opção **bug** da figura 10, abre-se a caixa do preenchimento do bug em que alguns campos estão pré-preenchidos, mas é o tester que escolhe qual o utilizador ou equipa que é destinado para correção do bug.

Para a criação de bugs existem os seguintes campos importantes que devem ser preenchidos:

- **<Enter title here>** - espaço para a nomenclatura do bug.
- **Assigned to** - para atribuir a equipa ou utilizador ao bug.
- **Area -** para escolher a área apropriada.
- **Iteration** - para escolher a iteração apropriada.
- **Priority** - indica a importância do bug (1 a 4).
- **Severity** - indica o impacto do erro no projeto (Critico, Alta, Media e Baixa).
- **Effort (Hours)** - para registar as horas imputadas para resolução do problema
- **Repro steps** - para fornecer detalhes, para o programador perceber melhor o problema.
- **Activity** - para indicar a atividade, sendo sempre testing para testers
- **Attachements -** para adicionar screenshots, fotos e imagens ao criar um bug.

Todos os campos de um bug são importantes, mas os campos **Priority** e **Severity** são fundamentais para se poder priorizar as correções. Por exemplo, o tester ao criar um bug com prioridade "**1**" está a informar ao programador o nível de prioridade da resolução do bug.

O programador, por sua vez ao consultar a lista de bugs que estão para ser corrigidos vai dar importância de acordo com o nível de prioridade do bug, ou seja irá começar a resolver os bugs com prioridade maior.

A **Severity** também é muito importante porque o programador consegue saber o grau critico ou a gravidade de um bug. Ou seja, quando o programador depara com um bug de grau critico ***(critical)*** dá mais importância à resolução do bug em relação a um bug com grau critico baixo ***(low).***

O tester de acordo com a sua experiência, vai poder classificar o grau crítico de um bug.

O **Repro steps** é um campo muito importante na criação de um bug, onde são descritos detalhadamente os passos de como reproduzir o bug, descrevendo o resultado esperado e o resultado obtido.

O objetivo deste campo é facilitar uma rápida compreensão do problema por parte do programador. A figura 11 seguinte mostra os principais campos de um bug antes do preenchimento pelo tester.

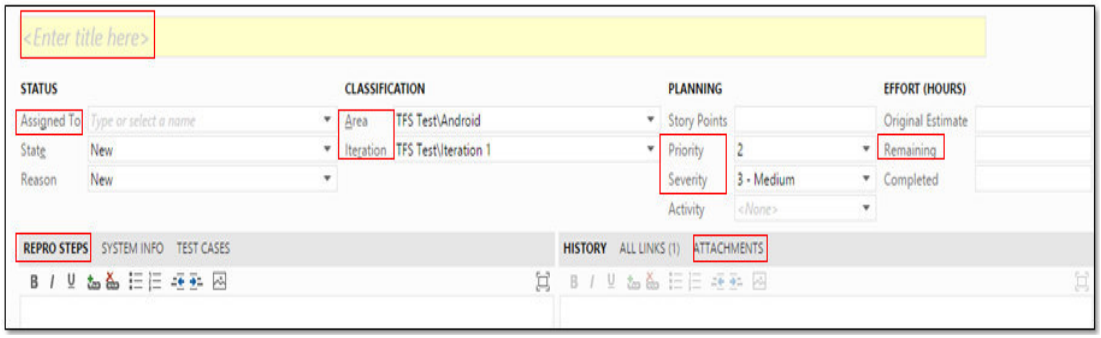

Ilustração 11 - Campos de um bug no TFS

A figura 12 mostra um exemplo de um bug com todos os campos preenchidos

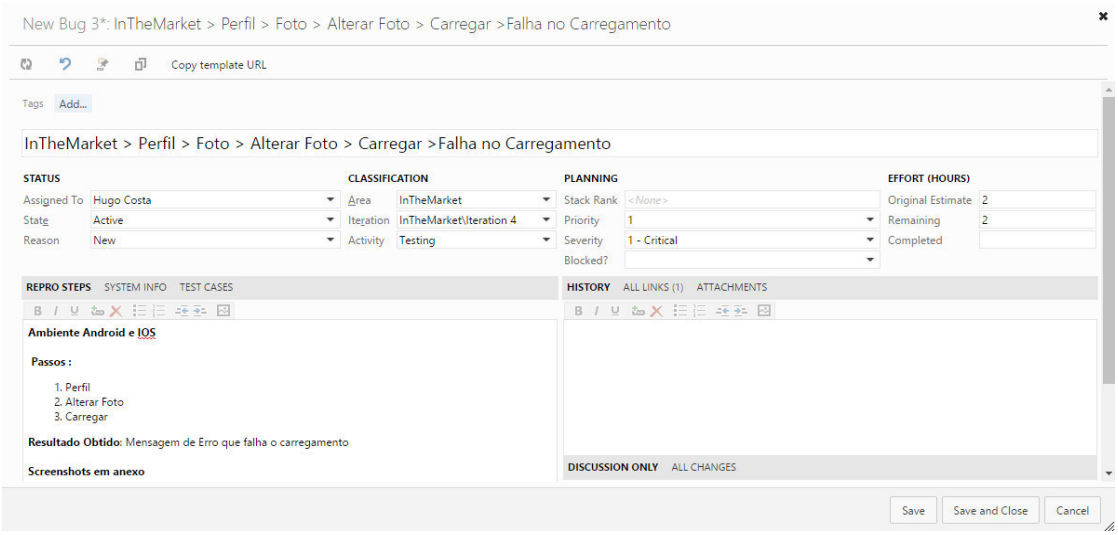

Ilustração 12 - Bug Preenchido no TFS

3.8.5 Bug Fechado

Como já foi dito anteriormente, um bug passa por vários estados até estar completamente resolvido ou fechado. A partir do momento em que um bug é criado, o TFS regista todas ações decorridas no bug, com registo de datas e hora em que ocorreram as ações bem como os utilizadores que alteram os estados. Quando um bug é resolvido os utilizadores têm a possibilidade de comentar a resolução do bug.

Existem algumas regras importantes definidas sobre a alteração do estado de um bug:

1. A abertura de um bug apenas pode ser feita pelos testers ou gestores do projeto
2. O programador apenas pode alterar o estado do bug para ***Resolved***
3. Apenas o tester pode alterar o estado para ***Closed***
4. Todos os bugs abertos devem ser confirmados com uma prova à sua resolução (Ex: screenshots)
5. Caso a resolução de um bug origine um novo bug, deve ser fechado o bug resolvido e abrir um novo.

A figura 13 mostra todo o estado de um bug corrigido e fechado.

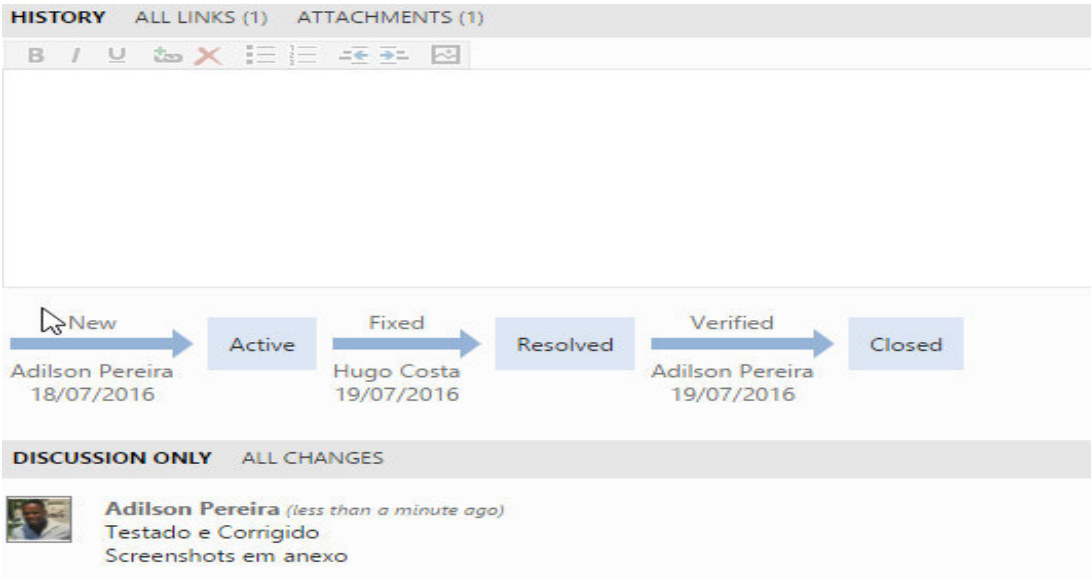

Ilustração 13 - Bug Fechado

3.8.6 Task

Uma ***task*** (tarefa) é uma tarefa ou funcionalidade criada para o programador fazer ou implementar. Normalmente é criada pelo chefe da equipa de desenvolvimento no momento da criação das iterações, mas também há certas alturas em que o tester pode criar uma tarefa desde que tenha experiencia suficiente para tal.

O tester no momento do teste, às vezes, depara-se com algumas funcionalidades que não estão presentes na aplicação ou que podiam ser melhoradas, podendo criar uma tarefa depois de consultar a opinião dos programadores ou chefe da equipa.

Uma tarefa depois de criada também pode ser removida pelo chefe da equipa se achar que não constitui melhoria para o projeto.

O workflow de uma ***task*** é semelhante ao de um bug, diferenciando-se em pequenos detalhes como o estado que é constituído por 3 fases: **New, Active e Closed**. Neste caso o programador pode, depois de completar a tarefa, alterar o estado para **Closed.**

A figura 14 mostra os estados de uma ***task.***

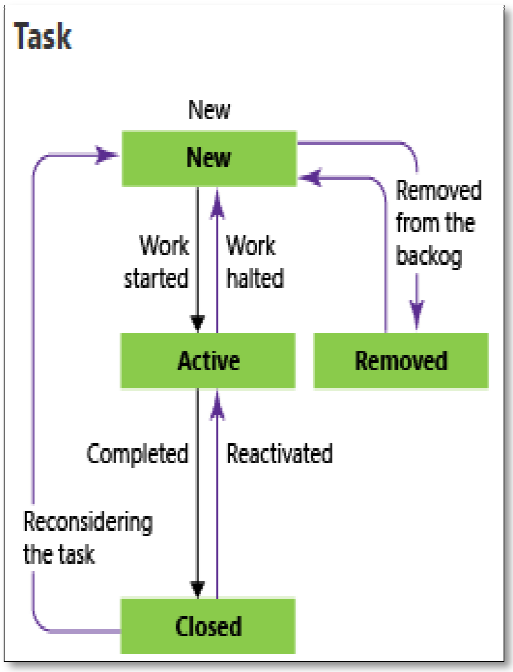

Ilustração 14 - Estados de uma Task

Em relação aos campos de uma ***task*** têm pouca diferença comparando com os do bug, sendo que não existe o campo ***Severity*****,** e em vez ter as "***steps***" tem a descrição que serve para descrever a tarefa, caso seja necessário. A figura 15 mostra o exemplo de uma ***task.***

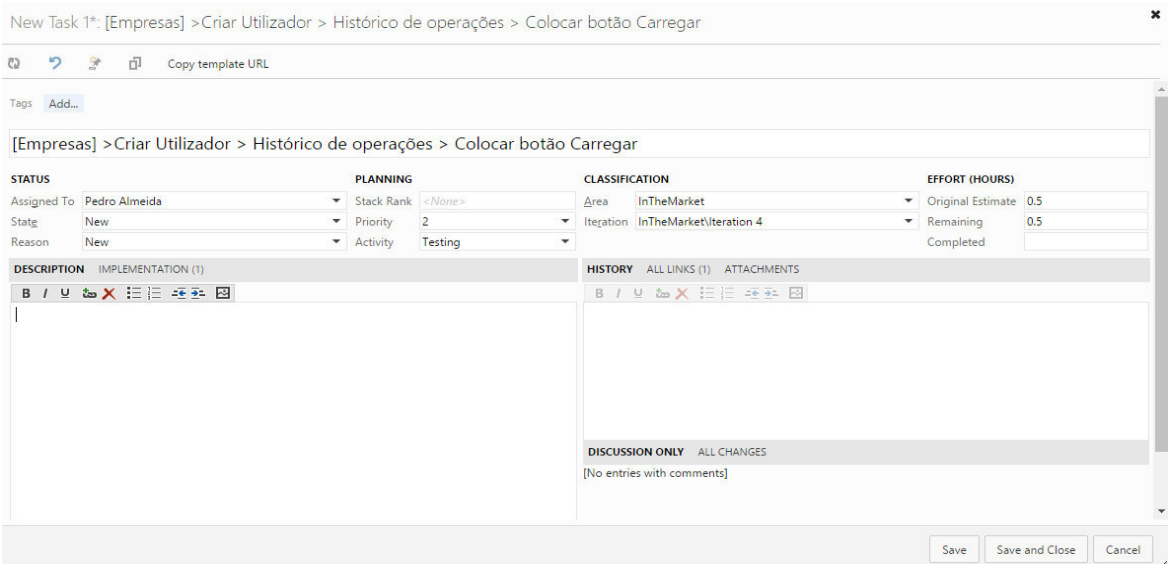

Ilustração 15 - Exemplo de uma Task

3.8.7 Sugestão de Melhoria

As sugestões de melhorias são também tarefas que o tester pode sugerir ao programador com o intuito de contribuir para o aumento da qualidade da aplicação. Normalmente para sugerir ação de melhoria abre-se um *task* e no título é indicado que se trata de uma sugestão de melhoria. A figura 16 mostra um exemplo de uma ação de melhoria.

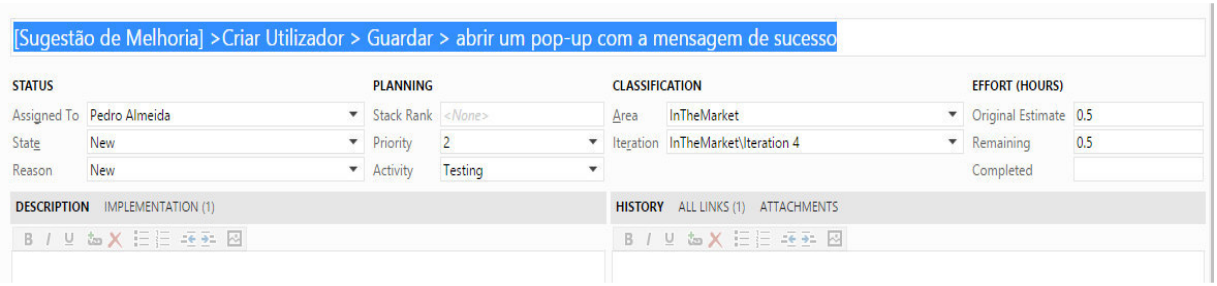

Ilustração 16 - Sugestão de Melhoria

4. Considerações Finais

4.1 Análise do Estágio

O estágio, alem do propósito principal para a obtenção do grau de mestre em Informática, especialização em sistemas de Informação, foi uma experiencia importante na área de testes. A empresa onde realizou-se o estágio tem como principal atividade, o desenvolvimento de aplicações inovadoras e novas soluções para atuais e potencias clientes. Com tudo isto o estagiário através da rápida integração foi-se inserindo em vários tipos de projetos importantes.

Os projetos desenvolvidos pela Itsector são na maioria soluções financeiras com grau de responsabilidade muito alto par parte dos desenvolvedores e por parte dos testers e isto permite aumentar o sentido da responsabilidade por parte do estagiário, sendo que os projetos são críticos como, por exemplo, as aplicações para bancos e seguradoras.

A forma como foi conduzido o processo de estágio foi uma mais valia para o enquadramento do estagiário na equipa de testes e a interação com a equipa de desenvolvimento. Os testes das aplicações eram subdivididos entre as equipas de testes e o estagiário, que como, estava num ciclo de aprendizagem, tinha sempre um tester experiente que acompanhava todo o trabalho desenvolvido. A supervisão do trabalho do estagio ocorreu apenas numa fase inicial, visto que, quando este adquiriu o conhecimento base, começou a ser o próprio responsável pelo seu trabalho, tendo uma interação diretamente com os programadores.

Em qualquer projeto em que esteve inserido, o estagiário tinha uma reunião semanalmente, com o gestor de projeto, o chefe da equipa e os programadores envolvidos para averiguar o ponto de situação da aplicação, em que o objetivo era simplesmente dar a conhecer a evolução do trabalho e as próximas tarefas a serem executadas.

O papel do estagiário ganhava mais importância, quando no final de cada teste de uma aplicação tinha de organizar um documento com a descrição e resultado de todas as funcionalidades testadas bem como as falhas corrigidas validando se a aplicação está pronta para a produção, caso confirmasse. Nenhuma aplicação passa a produção sem antes a aprovação do tester responsável pelos testes.

Este estagio serviu para adquirir novos conhecimentos, perceber a cultura organizacional de uma instituição e ganhar experiencia na área de teste na fase de desenvolvimento de software.

4.2 Principais Dificuldades

No início do estágio, sendo uma fase de adaptação o estagiário teve algumas dificuldades na integração com os colegas, mas aos poucos começou a integrar-se no grupo. Em termos técnicos, a comunicação foi uma das principais dificuldades apontadas na primeira reunião de estágio, ou seja, a forma pouca clara como o estagiário comunicava a existência de bug aos programadores era pouco percetível ou não estava descrita de uma forma muito clara. Foi um dos aspetos que lhe foi pedido para melhorar significativamente. A utilização da ferramenta TFS teve também algumas dificuldades no início.

Sendo a Itsector uma organização muito grande onde estão envolvidas muitas pessoas foi, logo no inicio, sugerido ao estagiário que melhorasse a forma de organização e os métodos trabalho, para simplificar o entendimento por parte de outros intervenientes.

No que diz respeito aos testes umas das principais dificuldades, no inicio relacionou-se com o preenchimento dos campos, o valor correto da atribuição do grau de importância na criação dos bugs, em especial na ***Priority*** e **Severity,** que foram ultrapassadas à medida que aumentava a experiencia.

A auto-dependência foi também uma das características de melhoria sugeridas pela responsável da equipa de testes no sentido de resolver os problemas pesquisando sozinho a solução.

No início dos testes das aplicações o estagiário teve algumas dificuldades relativamente a pouca utilização das aplicações desenvolvidas no seu dia-a-dia, visto que a maioria se tratava de aplicações financeiras, como bancos e seguradoras com funcionalidades críticas e um grau alto de responsabilidade; apesar de ter sempre o documento de análise disponível era necessário um conhecimento e experiência que só era possível com tempo por parte da pessoa a testar este tipo de aplicações.

De referir que, apesar das dificuldades identificadas acima, o estagiário tenteu sempre ultrapassar as adversidades, aprendendo com todos e melhorar, e com passar do tempo, adquirindo experiência e conhecimento.

4.3 Análise e sugestão para melhoria de processos de Testes na ITsector

O processo de testes da ITsector, apesar de ser bem aplicada, poderiam ser melhoradas consideravelmente caso se implementassem também os testes automatizados.

A ITsector aplica a técnica de testes manuais às aplicações desenvolvidas, esta técnica pode ser considerada de grande eficiência, mas a sua grande limitação é o tempo que consome para realização de uma simples funcionalidade e o facto de ser necessário praticar testes exaustivamente (Myers, 2004). O ciclo de teste é muito repetitivo, a cada atualização terão de ser repetidos de novo os mesmos casos de testes.

Relativamente aos custos dos testes manuais é mais baixo comparativamente à implementação de testes automáticos. No entanto, defende-se que, a implementação de testes automáticos tem mais custos no início do projeto, mas que quando corretamente implementados, diminuem o custo do processo de teste de software, reduzindo o tempo de execução de tarefas repetitivas (Bortoluci & Duduchi, 2015).

A implementação de testes automáticos não substituiria a execução de testes manuais, mais poderiam ser aplicadas as duas técnicas, considerando que nem todos os casos de testes podem ser automatizados.

O teste manual continuará a ser útil, por exemplo, para encontrar problemas de layout e bugs triviais, os quais não seriam encontrados através de testes automatizados. Através da técnica de testes manuais consegue-se identificar problemas de usabilidade que não podem ser descobertos através da escrita e execução de teste automatizados.

As tarefas de testes manuais dependem muito do juízo humano, um erro, uma distração no momento de validação dos testes poderá ter influencias na validação do teste de software.

Uma vez que a ITsector segue a abordagem de desenvolvimento ágil a implementação de testes automáticos é muito recomendada na utilização desta metodologia, seria muito importante na otimização do tempo e dos recursos no processo de testes.

É difícil nos testes manuais, por exemplo, testar uma aplicação com 100 ou mais utilizadores registados ou até a utilizarem ao mesmo tempo. Já com testes automáticos seria possível simular um grande número de utilizadores a utilizar o sistema.

Os testes manuais exigem mais recursos humanos em comparação com os testes automáticos. É importante referir e relembrar que a automação de testes serve para auxiliar e não para substituir o papel do software tester (Patton, 2001).

4.3.1 Benefícios da implementação de testes automáticos

A automação de testes tem como princípio aumentar a eficiência no teste de software, pelos aspetos chave de redução dos custos em teste de software, redução do tempo dispensado durante a fase de testes, melhoria da cobertura dos testes e repetição dos testes (Coutinho, 2015). Depois da sugestão da implementação dos testes automáticos apresentam-se alguns benefícios que a Itsector poderia obter com a implementação desta técnica:

- Capacidade de reutilização dos casos de testes, ou seja, tornava possível executar os mesmos casos de testes inúmeras vezes, reduzindo o esforço e tempo em comparação com os testes manuais.
- Em projetos dinâmicos como é o caso da maioria dos projetos de Itsector podiam ser, mais facilmente, repetidos os casos de teste a cada mudança.
- Aumentava a cobertura de testes, possibilitando o planeamento de testes muito mais extenso, isto é, executava mais funcionalidades na aplicação a ser testada validando mais pontos de validações em apenas uma única execução de teste.
- As atividades de teste de software podiam ser mais rápidas a cada *release de* software, porque podiam ser executados uma bateria de testes automáticos previamente desenvolvidos em apenas poucas horas, enquanto que nos testes manuais pode demorar dias.
- Os testes automáticos poderiam trazer mais confiabilidade em relação aos testes manuais, pois pode-se ter a certeza que serão sempre

executados apenas as funcionalidades e valores que foram especificados durante o desenvolvimento dos testes automáticos tendo em conta que me testes manuais

- Acabaria com a dúvida clássica *"será que tester não cometeu falhas durante os testes"*, como por exemplo inserção incorreta de valores, funcionalidades por executar ou erros dos testes.
- Em pouco tempo poderiam ser executados dezenas ou centenas de casos de teste e tornava possível simular centenas de utilizadores acendendo a um sistema.
- As ferramentas de automação permitiriam gravar o conjunto de testes e reproduzi-lo em qualquer momento, caso fosse necessário.

4.4 Conclusão

Depois de concluído o período de estágio na área de testes percebe-se a grande importância que a fase de testes têm no desenvolvimento de software. Aplicações são desenvolvidas todos os dias e mesmo depois de serem entregues aos clientes continuam a ser incorporadas ou alteradas novas funcionalidades e por isso torna-se necessário novos testes, daí se conclui que os testes são processos contínuos que acompanham sempre a aplicação.

Além disso, é importante realçar que diferentes tipos de aplicações possuem diferentes técnicas de teste, ou seja, cada tipo de aplicação possui características específicas que devem ser consideradas no momento da realização dos testes. O teste de um software não garante a sua total qualidade, mas contribui para aumento da sua qualidade e reduz a possibilidade de erros e falha.

Um ponto bastante importante para a viabilização da aplicação de testes de software é a utilização de uma infraestrutura adequada. Realizar testes não consiste simplesmente na geração e execução de casos de teste, mas envolve também questões de planeamento, gestão e análise de resultados.

O apoio das ferramentas para qualquer atividade do processo de teste é importante como mecanismo para redução de esforço associado à tarefa em questão, seja ela planeamento, ou execução dos testes.

O TFS é uma ferramenta muito importante na execução das tarefas de testes facilitando imenso o trabalho dos testers e desenvolvedores.

Para concluir o relatório acrescento que não existe nenhuma aplicação desenvolvida que dispensa a elaboração dos testes, por isso esta atividade é fundamental.

Apesar das dificuldades descritas considero que os objetivos foram compridos.

Bibliografia

Ammann, P., & Offutt, J. (2008). *Introduction to Software Testing* (1st ed.). New York: Cambridge University Press.

Anacleto, J. A. (2012). *Desenvolvimento de uma aplicação web para dispositivos móveis -*. Dissertação, Universidade do Minho.

Barbosa, E. F., Maldonado, J. C., Vincenzi, A. M., Delamaro, M. E., Souza, S. d., & Jino, M. (2010). Introdução ao Teste de Software.

Bortoluci, R., & Duduchi, M. (2015). Sistemas Produtivos e Desenvolvimento Profissional: Desafios e Perspectivas. *Um estudo de caso do processo de testes automáticos e manuais de software no desenvolvimento ágil*.

Carvalho, M. F. (2010). *Automatização de Testes de Software.* Relatorio Estágio, Instituto Politécnico de Coimbra Instituto Superior de Engenharia .

Coelho, T. F. (2016). *Automação de testes de aplicações móveis sem necessidade de programação.* Dissertação, Faculdade de Engenharia do Porto, Porto.

Cohn, M. (2004). *User Stories Applied.* Pearson education.

Collins, E. (17 de Setembro de 2013). Automação de testes.

Correia, S. A., & da Silva, A. R. (2004). Técnicas para Construção de Testes Funcionais Automáticos.

Coutinho, G. E. (2015). *O papel das ferramentas de automação de testes funcionais em contexto de projetos dinâmicos e complexos.* Dissertação , Universidade de Trás-os-Montes e Alto Douro.

da Silva, M. F., & Moreno, A. G. (2011). Automação em Testes Ágeis. *Revista de Sistemas de Computação*.

Devmedia. (2010). Qualidade de Softeare. *Engenharia de Software*.

Farias, E. (5 de Março de 2014). A arte do Teste de Software. *O que é teste de Software?*

Gregory, J. (2009). Focus Your Testing. *DragonFire Inc.*

Hillabin, J. (2012 de Abril de 2012). 6 Disasters Caused by Poorly Designed User Interfaces.

IEEE. (1990). *Standard Glossary of Software Engineering Terminology.*

ISTBQ. (s.d.). *EXAM CERTIFICATION*. Obtido em 02 de Fevereiro de 2017, de Why is software testing necessary?: http://istqbexamcertification.com/why-is-testing-necessary/

Itsector. (2012). Guia de desenvolvimento do Itsector.

Itsector. (2016). Obtido de http://www.itsector.pt/pt/quem-somos/

Maderna, A. (Abril de 2016). *Kanoah Test*. Obtido de Why is Testing Necessary?

Marszałek, K. (31 de Março de 2016). *Functional Testing Best Pratices*. Obtido em Junho de 2016, de How we do startups: http://howwedostartups.com/articles/functional-testing-best-practices

Myers, G. J. (2004). *The Art of Software Testing* (2rd ed.). New Jersey: John Wiley & Sons, Inc.

Neto, A. C. (2007). Introdução a Teste de Software. *Engenharia de Software Magazine, 1*.

Patton, R. (2001). *Software Testing*. Indiana, USA: Sams Publishing.

Pinheiro, S. A. (2015). *Estudo e implementação de testes de software em desenvolvimento ágil.* Universidade do Minho, Braga.

Pressman, R. S. (1995). *Engenharia de Software* (1 ed.). Makron Books.

Ribeiro, G. F. (27 de janeiro de 2012). *Profissionais TI.* Obtido de Erros de Software: https://www.profissionaisti.com.br/2012/01/alguns-dos-mais-famosos-erros-de-softwares-da-historia/

Shiralige, A. (2016). *Agile Buddha.* Obtido de Agile Testing is achieving quality in everything: http://www.agilebuddha.com/scrum/agile-testing-is-achieving-quality-in-everything/

Singh, Y. (2011). *Software Testing* (1st ed.). Cambridge : Cambridge University Press.

Sommervile, I. (2011). *Software engineering* (9 ed.). Pearson Education, Inc.

Sommerville, I. (2015). *Software Engineering* (10 ed.). Pearson.

Zanin, A. (13 de Abril de 2011). *A importância dos testes de sofware*. Obtido em Fevereiro de 2017, de https://www.profissionaisti.com.br/2011/04/a-importancia-dos-testes-de-software/

Anexo

Calendarização do Estágio

Atividades	**Data**	**Tarefa**
Integração	18/01 - 23/01	Tutorial TFS
Testes	28/01 - 05/02	Reportar Bugs
Testes	05/02 – 19/02	Reportar Bugs e Relatório de Evidencias
Testes	22/02 – 26/02	Reportar bugs
Testes	29/02 – 04/03	Reportar Bugs
Testes	07/03 – 11/03	Reportar Bugs
Testes	14/03 – 18/03	Reportar Bugs
Testes	21/03 – 25/03	Reportar Bugs
Testes	28/03 – 01/04	Reportar Bugs
Testes	04/04 – 08/04	Testar e Validação
Testes	18/04 – 22/04	Testar e Validação
Testes	25/04 – 29/04	Testar e Validação
Testes	02/05 – 06/05	Testar e Validação
Testes	09/05 – 13/05	Testar e Validação
Testes	16/05 – 20/05	Testar e Validação
Testes	23/05 – 27/05	Testar e Validação
Testes	30/05 – 03/06	Testar e Validação
Testes	06/06 – 10/06	Reportar Bugs
Testes	13/06 – 17/06	Testar e Validação
Testes	20/06 – 24/06	Testar e validar
Ausente/ferias	27/06 – 01/07	
Ausente/ferias	04/07 – 08/07	
Testes	11/07 – 15/07	Testar e validar
Testes	18/07 – 22/07	Testar e validar
Testes	25/07 – 30/07	Testar e validar